Section 6

Uniform Data Handling Without Common Formats
(Apr 1992)
Uniform Data Handling Without Common Formats

This section has a few readings about how to handle the question of formats for data storage and data transfer.

• A talk about formats: Common Data Formats: Common Data Problems, Jan. 1992

• Improving Data Quality with an Interactive Data Workbench, Mar. 1992

This text describes methods to provide uniform data access even if formats are different.

• IPC Format Facility (Navy format conversion, 1992)

• The Myth of the Universal Meteorological Data Format (or One Size Fits All?)

• Data Management for JGOFS: Theory and Design, Feb. 1992

This text describes methods that accommodate different formats but gives a uniform view of the data.

• HP’s New Wave for Windows (sharing data between programs), Apr. 1992

• Upgrade Update - Windows 3.1 (Apr. 1992)

Windows version 3.1 has DDE (dynamic data exchange) and OLE (object linking and embedding). I still do not know exactly how this works.

- End -
A talk about formats:

“Common data formats: Common data problems?”
by R.E. Haberman, National Geophysical Data Center (NGDC);
and D.R. Mock, NOAA/ERL/CMDL,
both in Boulder, CO.

At NGDC, they face the usual problem that various projects collect data from diverse sources, but they want to make the data available for use on common software. The authors noted that there are two ways to handle this problem:

1. Put all data into standard formats
2. Use smart programs that interface between the archive format and the software

They created a data format description language which is easy to use and flexible.

- Must be flexible
- Must be easy. Scientists do not want to spend unnecessary time to accomplish this data import.

They also created a library of functions for the interpretation of this data description language.

Goals:
- Be easy to use
- Easy to maintain and extend
- Allow easy access to many existing data sets in existing formats
- Be fun.

Data objects

- Bring modern data access (independent of formats) to intermediate-level programmers
- The creation and manipulation of objects is easy
- The amount of overhead is small
- Data caching is built in to the procedure

This is written in C and has been ported to Macs.
A comment: Someone told me that this approach does not work well across systems. But it is being used on some different systems.

New developments
They are exploring the use of dynamic data - link languages

Where was the paper given:
1. At Annual AMS meeting Jan 5-10, 1992
2. Session on Interactive Information Processing Systems
3. The paper is in the preprints for the session.

- End -
Improving Data Quality With An Interactive Data Workbench
Ray E. Habermann
National Geophysical Data Center
(Handles Data in Native Formats)

Project Description

The NOAA Data Centers receive a large amount of data from a diverse group of contributors. The Data Centers have long been involved in distributing these data to the research community; however, the quality of many of these data sets has never been examined, even in the most cursory way. The principal obstacle to such an examination is the variety of formats in which these data exist.

We have recently made great progress toward breaking the data format barrier with the development of the FREEFORM data access system. This system was conceived at NGDC and has been used extensively for data preparation and quality control. The cornerstone of the system is an extremely simple data description language which gives the name, position, and type of each variable in the data file. The programs which take advantage of this system are able to access the data using these descriptions and, therefore, are effectively independent of the format of the data.

The FREEFORM system has been developed with the goal of being able to read data in the format which it comes to us, the "native" format. This means that a large portion of the data at NGDC can be described using this language and accessed using this system without changing the data. This is the real power of the system. We can create standard quality control procedures without standardizing the data.

These developments allow us to institute a generalized data quality control system which can be applied to a large portion of the existing data at the National Geophysical Data Center (as well as the other NOAA Data Centers) and to new data as it arrives. This system will form the foundation for systematically improving the quality of the data distributed by all of the NOAA data centers.

Many parts of the system already exist and have been tested and used successfully at NGDC. Programs exist for creating both text and graphic summaries of the data in a particular file.

Describe the project proposed quality and productivity improvements.

It has already become clear that a unified data access system can lead to great increases in productivity. The data managers can concentrate on the data itself, rather than the format of the data. We have used the FREEFORM system to systematically examine data sets which have been in the data center for years and have never been accessed before because no one has had the time to write the format specific programs for reading those data sets. Now standard programs can be used to access all of the data sets, regardless of their formats.
Chapter 6

1 IPC FORMAT FACILITY

1.1 Introduction

In the preceding chapters, emphasis has been placed on the mechanics of how to exchange information between client and server applications. However, just as important is the content of the information being exchanged.

One can view the IPC as a catalog of information that can be accessed at an application's leisure. The information can be "ordered" and, more often than not, must be assembled by the end user once received. An instruction book is necessary to allow the consumer (client) to understand and assemble the product (information) they have received.

To accomplish this, the IPC provides a facility that allows a server to associate a data format with a data item. The user is free to create their own data format definitions, however, without standardization this would be useful only for IPC products using the user-defined data formats. Therefore, the IPC provides a data format facility that can be used by all IPC applications to provide a standardized method for encapsulating data content.

In the IPC, a data format is called a format resource because it is essentially an object that can be used by an application as many times as necessary. A format resource can be thought of as a scripted language that allows one to describe intricate data formats in a concise manner. A format resource can be created using the IPC Format Resource Compiler. The IPC Format Resource Compiler creates a binary resource file that can be loaded by a server at runtime and assigned to various data items or request topics.
The data format facility is also useful for transferring data between different computer architectures. It is used by the IPC network agents to provide communications of data between such diverse machines as the Sun Workstation, IBM-PC, VAX, and Cray. The IPC data format facility fits into the ISO presentation layer and is roughly analogous in purpose to X.409, ISO Abstract Syntax Notation.

The following list summarizes some of the advantages of using the IPC Data Format Facility:

- **Isolates** the users of information from the underlying method of storing information.
- Provides a method of assigning ASCII names to data fields, allowing the user to directly interact with the flow and content of information.
- Supports common data types (integer, floating point), fixed data types, nested arrays, nested records, and bit fields.
- Supports unit assignments and unit conversions (with predefined conversions for angular, distance, weight, volume, and velocity measurements).
- Provides a vehicle for the transfer of binary data between machines using different data orientations (Motorola-Big Endian, Intel-Little Endian).

#

In original text: "There are 31 pages in Chapter 6,"
Navy Perfects Dial-Up Weather System

A PC tells which way the wind blows

By BOB BREWIN

Modern mariners, like their ancient forebears, still have to keep one eye out for the weather, but Navy forecasters now can use PCs to tap into a supercomputer eye on a storm. Navy and other Defense Department forecasters can access the same revolutionary, PC-based weather forecasting system to draw down Defense Meteorological Satellite Program imagery as well as 3-D ocean data.

By tapping into the global weather database, which is maintained on a Control Data Corp. Cyber mainframe at the Fleet Numerical Oceanography Center, Monterey, Calif., users of the PC-based Naval Oceanography Distribution System (NODS) can customize a forecast to run from the extreme depths of the ocean, such as the seven-mile-deep Marianas Trench, to more than 10 miles into the Earth's atmosphere.

John Garthner, Routing Services Division head at Fleet Numerical, said the program is so easy to use that even the computerphobic can master it quickly. "We have more than 350 DOD users on-line. We just issue them a manual. No one has ever requested or received formal training, and within two hours anyone can use NODS proficiently."

NODS is a menu-driven system. Users select the geographic area they want to examine from a list and pick from 70 weather products available to download from the supercomputer: fronts, storm warnings or a variety of bathymetric data so important to the submarine and anti-submarine warfare community.

With Return, and NODS does the rest. It dials into Monterey through a number of gateways -- including Tymnet, the Defense Data Network, the Defense Switched Network or commercial phone lines -- and interrogates the computer core, which compresses the data. NODS automatically unpacks the data, which is ready within minutes to be called up and manipulated by the user.

NODS is a revolutionary tool, according to Cmdr. Larry Warrenchefutz, meteorology programs branch head at the Office of the Oceanographer of the Navy, Washington, D.C. "You can do things with NODS you could never do before or could do only with great difficulty," he said.

The ability of NODS to overlay forecast data onto satellite imagery and then "loop" the screen and run through days' worth of imagery and forecasts gives DOD forecasters a valuable and timesaving tool.

"Before you had to deal with satellite images in one scale, charts in another scale and weather maps in yet another. You'd have all these pieces of paper lying around and have to eyeball how one correlated with the other," he added.

Warrenchefutz said he also likes the fact that NODS is a user-driven "pull" system rather than a provider "pushed" system. "We still do fleet weather broadcasts 24 hours a day, but you have to make sure you are around when the broadcast for the part of the world you're interested in comes down. With this, you select the area and products you want, and the computer gets them for you."

This is the dial-up PC weather map system being developed by Doug McLain and others. It's an FNOCS system. They are making it available to civilians. They use NEON's (display system) as a test for NODS.

NODS = Naval Oceanographic Data Distribution System

Global Weather Info

Through the Naval Oceanography Distribution System, users at sea or stationed around the world can access a global weather database to get custom forecasts.

WEATHER, FROM PAGE 510

NODS is a revolutionary tool, according to Cmdr. Larry Warrenchefutz, meteorology programs branch head at the Office of the Oceanographer of the Navy, Washington, D.C. "You can do things with NODS you could never do before or could do only with great difficulty," he said.

The ability of NODS to overlay forecast data onto satellite imagery and then "loop" the screen and run through days' worth of imagery and forecasts gives DOD forecasters a valuable and timesaving tool.

"Before you had to deal with satellite images in one scale, charts in another scale and weather maps in yet another. You'd have all these pieces of paper lying around and have to eyeball how one correlated with the other," he added.

Warrenchefutz said he also likes the fact that NODS is a user-driven "pull" system rather than a provider "pushed" system. "We still do fleet weather broadcasts 24 hours a day, but you have to make sure you are around when the broadcast for the part of the world you're interested in comes down. With this, you select the area and products you want, and the computer gets them for you."

This is the dial-up PC weather map system being developed by Doug McLain and others. It's an FNOCS system. They are making it available to civilians. They use NEON's (display system) as a test for NODS.

NODS is a revolutionary tool, according to Cmdr. Larry Warrenchefutz, meteorology programs branch head at the Office of the Oceanographer of the Navy, Washington, D.C. "You can do things with NODS you could never do before or could do only with great difficulty," he said.

The ability of NODS to overlay forecast data onto satellite imagery and then "loop" the screen and run through days' worth of imagery and forecasts gives DOD forecasters a valuable and timesaving tool.

"Before you had to deal with satellite images in one scale, charts in another scale and weather maps in yet another. You'd have all these pieces of paper lying around and have to eyeball how one correlated with the other," he added.

Warrenchefutz said he also likes the fact that NODS is a user-driven "pull" system rather than a provider "pushed" system. "We still do fleet weather broadcasts 24 hours a day, but you have to make sure you are around when the broadcast for the part of the world you're interested in comes down. With this, you select the area and products you want, and the computer gets them for you."

This is the dial-up PC weather map system being developed by Doug McLain and others. It's an FNOCS system. They are making it available to civilians. They use NEON's (display system) as a test for NODS.
NIH wasted millions on computers

Washington
In 1988, computer officers at the U.S. National Institutes of Health (NIH) contracted with IBM for a major computer system with a potential value over ten years of $806 million that was meant to meet all of the institutes' computing needs. Unfortunately, those responsible for deciding to go with an IBM mainframe system failed to ask NIH scientists what kind of computing systems they wanted. Had they done so, according to a report from the US General Accounting Office, they would have discovered that many of the scientists favoured plain PCs and have no need for the "total system" for which NIH issued a contract.

The GAO report, which current NIH officials have not contradicted, records a priceless example of bureaucracy run amok. Although the NIH have various computer advisory committees, the bureaucrats who chose the IMB mainframe system failed to consult any of them. Nor, according to the GAO, did they interview scientists because "they did not believe it was worthwhile to spend time surveying scientists," in part apparently because of a survey 20 years previously that "produced unmanageable results".

The total system NIH now has not only has excess capacity, but also includes one full-sized computer dedicated to backup and related needs. However, GAO notes that NIH mainframes (leased in earlier years) are connected in such a manner that one backs up another during a system failure. "Consequently, NIH mainframes virtually never fail and have been so reliable that NIH no longer keeps mainframe failure data."

As to the scientists, who were heard from through an advisory body 4 months after the 1988 contract was signed, the availability of new software for personal computers rendered the need for mainframes for most scientific research outmoded. Besides, they said, the user charges for the multi-million dollar IBM system were too high.

So, while most researchers are happy with their PCs, NIH has wasted millions on a system is apparently does not need. GAO recommends that in the future NIH should solicit data on scientists' needs before buying them equipment they do not need and cannot afford. The current NIH leadership admits that is pretty good advice.

Barbara J. Culliton
The Myth of the Universal Meteorological Data Format
(or One Size Fits All?)

by

Joseph P. Reuden and John S. Pyeatt

University of Wisconsin - Madison
Space Science and Engineering Center

Abstract

The current emphasis within various meteorological programs is to declare a single data format sufficient for all data needs with only cursory provision made for different data types. However, no provision is generally made for performance, resource consumption, and on-time needs.

SSEC has been developing and evaluating meteorological data formats for over 15 years and has built up considerable expertise in this area. This paper will explore and describe various tradeoffs and their interrelationships which must be balanced to provide robust, effective data format systems.

Joseph P. Rueden
1225 West Dayton St.
University of Wisconsin - Madison
Madison, Wisconsin 53706

Phone: 608-263-3897
FAX: 608-262-5974
Email: joer@ssecmail.ssec.wisc.edu

Background (3 June 1992)

J.T. Young told me (Roy Jenne) about this coming paper that will be presented in the Jan. 1993 meeting of the American Meteorological Society. J.T. Young and Joseph Rueden manage the development and operation of the well-known McIDAS display system at the University of Wisconsin. McIDAS is used at many locations in the U.S. and overseas.

Data Compression at the University of Wisconsin

For some tasks, J.T. Young says that the University of Wisconsin has been using data compression of satellite images since the mid-1970s. At that time computer memory was very expensive, and they usually achieved 3:1 compression of the volume.

Now the University of Wisconsin often uses loss-less data compression for data transmission. They usually achieve about a 4 to 1 compaction of the volume. It is a modified delta scheme that should be quite efficient in its use of computer time. Many users obtain images over relatively slow 9600 bit-per-second transmission lines, and this compression helps a lot. Compression has not been used for the main satellite archives.
THE MYTH OF THE UNIVERSE METEROLOGICAL DATA FORMAT
(OR ONE SIZE FITS ALL?)

Joseph P. Rueden and John S. Pyeatt

University of Wisconsin - Madison
Space Science and Engineering Center

September 16, 1992

1. INTRODUCTION

The current emphasis within various meteorological programs, and scientific programs in general, is to declare a single data format sufficient for all data needs and uses. This is done in the hope it will simplify the management of the program and facilitate sharing of the data between participating parties. But single data formats have substantial hidden costs. This can become a driving force in determining computing resources, connectivity, and administrative requirements; other programs pay handsomely for the data to be converted to their (undoubtedly different) format; and multiple formats frequently wind up being supported late in the evolution of the program as changing requirements expand the community with which data must be interchanged.

This paper will discuss what factors traditionally have come into play when deciding on data formats, why a single data format is insufficient, and what steps the community needs to take to improve the current situation.

2. CURRENT METHODOLOGY

Programs today generally have a study done to determine what the 'right format' is for this program. Managerial direction is usually given regarding transportability, off-the-shelf availability, and occasionally applications availability. Almost never does performance, maintenance, or administrative effort become an issue at this point.

A technical group then does a study of a variety of formats weighing the factors they were directed to, adding various technical issues (like cpu, storage, or transmission performance and suitability for the expected data types), and generates a mostly subjective measurement of the various factors. Early reports from the study group invariably indicate that none of the data formats studied meet all the criteria, although some formats handle subsets of the criteria at an acceptable level.

By now agreements must be reached between the various participating groups. So, without a strong technical recommendation and with tremendous pressure to get on with it, one or two formats are chosen by political consensus.

Finally, the study paper is written, and (surprise!) it supports the selection which was made politically regardless of the supporting study data.

3. WHY DOES THIS HAPPEN?

Data formats have high visibility in this era of standards and are considered part of the interface between participating groups that must be specified early on in a program. Because of this high degree of visibility, the complex issue of data formats quite often ends up being a politically motivated decision.

4. ONE SIZE DOES NOT FIT ALL!

One size does not fit all is because the spectrum of computing platforms, communications, data types, and intended uses is too broad and changing too rapidly to permit efficient system implementation using only one format. Good overall system performance, which includes the users' and administration time, is essential to keep costs low over the life of a program.

4.1 Computing Platforms

The large variation in platform profiles and attribute uses mandates different format solutions to guarantee timely data access. This one is the easiest to understand and measure.

A mainframe environment (multi-channel, timeshared) is very different from a workstation environment (generally RISC-based processor, single channel, typically shared among a couple of people) which is in turn very different from a personal computer environment (Intel x86 or Motorola 68xxx type processor, single channel, single user). The chart below lists relative performance.
The key reason for this is because more data compression employed by a format translates into more system resources, possibly I/O and certainly CPU cycles, necessary to put the data in memory access format. This problem is further multiplied when the spectrum of computing platforms used in the scientific community is taken into account.

Will the trends in computer technology make data format efficiency unimportant? Probably not. Observational systems currently being developed will create larger data sets than ever before. Users will also be putting more strain on these computer resources as their expectations grow with the technology.

4.2 Communication Capabilities

A very wide range of communication bandwidth is in use today: operating communication speeds range from 50 bits per second to at least 100 MBytes per second. Data formats designed for the very slow communication lines typically stress the absolute maximum in compression, those in the upper range typically stress minimizing delays in feeding the data to the communications system, and those in the middle ranges must balance the various factors to achieve acceptable overall performance. Because many programs handle a variety of communications at a variety of speeds, one format cannot be cost effective.

4.3 Data Types

The form of the data itself is a well recognized concern. Data can be categorized, but not all data fits a category well. Common categories today are uniform grids, imagery, text, and in situ or point source data.

Considerable effort has gone into design and evaluation of data formats with regard to different data types. Desirable attributes include ability to contain the data properly and completely, cohesiveness, extensibility, storage and processor efficiency, etc. This is not enough! The other factors listed here must also be considered.

4.4 Intended Data Uses and Access Patterns

The intended use of the data (the access patterns) and timeliness requirements of the user are key factors which must be considered in the design of a data format. Examples of these sorts of criteria are:

### | Pcs | Workstations | Mainframes | Usage | Attributes
--- | --- | --- | --- | --- | ---
ClibPDF - www.fastio.com

| | Low | Low | Low | Low | Pcs |
--- | --- | --- | --- | --- | ---

| | Modest | High | High | High | Workstations |
--- | --- | --- | --- | --- | ---

| | High | Low | High | High | Mainframes |
--- | --- | --- | --- | --- | ---

| | Processing | Processing | Processing | Processing | Usage |
--- | --- | --- | --- | --- | ---

| | Storage | Bandwidth | Memory | CPU | Attributes |
--- | --- | --- | --- | --- | ---

The data formats implemented for meteorological programs to date have been able to handle a few, but not all, of the usage issues. As an example, while a data format that has a high degree of compression may be excellent for data transfer, this format will invariably be unacceptable for real-time, operational, forecasting environments.
- Realtime access required. The more stringent the timeliness requirement, the stronger the argument for multiple formats. This requirement frequently means accessing data before an entire set is collected, as in the first part of an image or the first few weather reports, even though the rest of the data is in the process of being received.

- Extensive vs Intensive data manipulation. Extensive data users generally need fast access to large volumes of data such as those used in displaying a satellite loop. Intensive data users generally need very fast computing on a modest amount of data such as the generation of vertical soundings from multi-channel satellite imagery.

- Interactive vs batch. Interactive processing has much more stringent timeliness requirements than batch because the user is waiting for the answer. The trend today is toward more and more interaction with larger and larger amounts of data.

- Archive and Retrieval. The primary requirement here is generally storage efficiency. Access rates increase in importance as we move to near-line storage with interactive search and order systems allowing users better access.

4.5 Other Factors

The real bottom line in any system is user satisfaction. If the data format substantially interferes with the systems ability to perform to the users’ satisfaction, then another format should be considered.

An example of this is the issue of serial vs random access formats. Random access formats are much more efficient on random access storage, e.g. disc, than are serial formats assuming the data needs to be randomly addressed, such as imagery. In general, serial formats are good on serial storage such as tape, or on low speed communications where the data is not expected to be randomly addressed.

Whether the data is a stable set or continuously growing is an example of an attribute that is not addressed by the above categories. When the data set continuously grows the subsystem encoding the data into the database format must have adequate performance.

5.0 WHAT CAN BE DONE?

What we propose is:
- Develop an objective method of measuring format efficiency. One way to do this is with the development of benchmarks that run on a variety of platforms, exercise data access in a variety of ways representing different intended data uses, and have one or more standard APIs the format developers must meet to run the tests. This would allow direct intercomparison of various formats. The differences in performance in different environments should then become much more obvious.

- Encourage development of translators, supporting the evolution of several of the formats in existence today. The goal of this is to be able to translate data from one form to another appropriate for the application desired. If you carry this to its logical conclusion, this would cause the different formats to evolve into a suite of compatible formats.

6. CONCLUSION

Significant positive change has occurred in the last 10 years. There is now general recognition of some major data types: grids, in situ or point source data, images, and text. There is strong desire among the programs to make their data accessible to other programs. Several data formats are competing for recognition as standards and significant effort is being devoted to improve those data formats. Finally, some programs are merging formats and others are building translators to convert between the various common data formats.

What’s needed now is recognition among the technical community that different formats are needed for different technical environments, that these environments are continually changing, creation of an accepted objective measurement system, and the formulation of systems specifically designed for handling a variety of formats. Only then can the answers to the technical questions regarding data formats be derived in a technical manner.
Data Management for JGOFS: Theory and Design

Glenn R. Flierl, MIT
James K.B. Bishop, LDGO
David M. Glover, WHOI
Satish Paranjpe, LDGO

Introduction

The Joint Global Ocean Flux Study (JGOFS), currently being organized under the auspices of the Scientific Committee for Ocean Research (SCOR), is intended to be a decade long internationally coordinated program. The main goal of JGOFS is "to determine and understand on a global scale the processes controlling the time-varying fluxes of carbon and associated biogenic elements in the ocean, and to evaluate the related exchanges with the atmosphere, sea floor and continental boundaries." "A long-term goal of JGOFS will be to establish strategies for observing, on long time scales, changes in ocean biogeochemical cycles in relation to climate change." Participation from a large number of U.S. and foreign institutions is expected.

JGOFS investigators have begun a set of time-series measurements and global surveys of a wide variety of biological, chemical, and physical quantities, detailed process-oriented studies, satellite observations of ocean color and wind stress, and modeling of the bio-geochemical processes. These experiments will generate data in amounts unprecedented in the biological and chemical communities; rapid and effortless exchange of these data will be important to the success of JGOFS.

Microcomputers and workstations have dramatically altered the gathering and analysis of oceanic data. While the convenience and ease of use of these machines make them ideal for an individual working on his or her data, the process of exchanging data or collecting relevant information from archived data sets is still difficult and daunting. Everyone uses different formats with different procedures for manipulating data;

Envision being able to sit at a microcomputer and ask what sets of phosphate data are available and then, based on the reply, ask for some suitable subset. The data could be imported to your local storage where it would arrive in the format you are accustomed to using for your own data or it could be used directly for creating a plot or as part of a calculation. Essentially, the JGOFS distributed data archive, as well as large amounts of historical data, would appear to be an extension of one's own data base — as readily available and as familiar in structure.

The storage format is likewise the PI's choice. Others can access the data without regard to storage method or location.

Many data base programs exist for small machines and, in and of itself, there is little value in developing another one. Rather, we have begun building an "object-oriented" (to be defined in more detail below) data base which has many unique features making it especially suitable for experiments such as JGOFS and WOCE. For the new initiatives on Global Change, data management will likewise be of fundamental importance (Natl. Acad. Sci., 1991). Systems such as ours provide the flexibility to handle such widely diverse data sets from many different sites and to integrate the information into useful form.
HP’s New Wave for Windows (sharing data between programs)

CATCH THE WAVE

While NDW augments Windows, Hewlett-Packard’s NewWave completely renovates it, taking Windows’ strengths and building on them. Upon seeing NewWave’s object-oriented front-end, some users might not even be able to tell they’re looking at a program based in Windows.

NewWave does more than just put a new “object” coat on Windows, though. Its Object Management Facility (OMF) brings the dream of transparent data exchange to life. Windows’ Dynamic Data Exchange (DDE) isn’t that strong. All DDE does is describe a way for applications to communicate with each other. Whether a program could make sense of the DDE-transmitted information was up to that software’s designers.

Hewlett-Packard’s OMF makes it possible for dissimilar programs to share information in realtime without the manual labor required by DDE. You could, for instance, make a change to an image in your paint program and have the change reflected by its linked illustration in your desktop publishing program. In this situation, NewWave would run the DTP in the background. With OMF, you can even move or rename the image-holding file and the data links would remain intact.

There are limits, of course. NewWave can do its magic only with NewWave-compatible or “encapsulated” DOS or Windows programs. Encapsulating a program requires the use of NewWave’s bridge program. Even so, NewWave can pull this stunt off with more programs than Windows can with DDE.

If you keep up with computer news, however, you’ll know that Microsoft has an addition to Windows called Object Linking and Embedding (OLE) that will do all that HP’s OMF can do and more. This doesn’t mean that you should abandon NewWave. Hewlett-Packard and Microsoft are working together to ensure that Windows 3.x with OLE will be compatible with NewWave, but Windows version 3.1 (hopefully out as you read this) will include OLE support.

Another feature of NewWave that makes it stand out from the crowd is its Agent functions. This program is a glimpse into computing’s future. While you could simply call it a macro recorder and runner, Agents are far more than jazzed-up batch files or script programs. Used properly, Agents can automate much of your computer grunt work. Their flexibility and ability to run programs—and commands within programs—at set times makes them have almost limitless possibilities.

The one bad thing about NewWave is that it craves system resources. Don’t even bother to think about running it on anything less than a fast 386 with a matching 60Mb-plus hard drive.

The wave of the future may indeed be NewWave. It can use Windows programs, there are a small but growing numbers of NewWave-specific applications, and HP has shown every willingness to support NewWave until it catches on. Besides that, HP is porting NewWave to Unix and there are already existing NewWave groupware products like AT&T’s Rhapsody and HP’s own NewWave Office. Despite rave reviews, however, NewWave has never really caught on. Nevertheless, as more users have systems that can do NewWave justice, that may change.

APRIL 1992 • COMPUTER SHOPPER
Upgrade Update

Anticipation of Windows 3.1 Spurs DDE and OLE Support

By Carol Ellison

Products highlighted in this month's update reflect the software industry's anticipation of Windows 3.1. Upgrades designed to exploit the Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE) capabilities of Windows 3.1 hit the market this winter in advance of 3.1 itself. Microsoft's own Project for Windows 3.0 and Ventura Publisher 4.0 for Windows shipped in January with DDE and OLE support. And Computer Associates, looking to the future, announced that future versions of SuperCalc, though not due for another year, will support both features.

DDE is not new to Windows users. Its powerful data linking—allowing different applications to call data from other applications and automatically update their own—was present in Windows 3.0. But OLE comes into its own in version 3.1. In essence, it brings some of the advantages of integrated software to stand-alone applications. OLE allows a user to embed objects in a document to data stored in other applications—launching those applications when you click on the embedded object.

Even before its release in January, Microsoft Project for Windows 3.0 was promoted as pricey-but-worth-it, with an upgrade price set at a healthy $199. Right after the holidays, reviewers and beta testers began applauding its muscle. With its multiple views and windows that users can open, Windows seems to be just the environment for planning and organizing tasks in multifaceted projects.

Microsoft actually skipped a version number in releasing Project for Windows (jumping from version 1.0 to 3.0) to keep the product parallel with its Macintosh version and to suggest just how much of an upgrade users were getting. Version 3.0 implements both DDE and OLE to support live links to other documents and allows users to bring the functionality of other applications to their projects. The new version also features a print-preview mode that lets you view multiple pages at once, a boon for those wanting to view a full layout of complex Gantt and Pert charts before printing. Project also boasts its own phone number for sales and upgrade information, (206) 635-7155.
Section 7

Two Items Toward an Analysis of Formats
To: Roy Jenne
From: Jill Travers
Date: December 13, 1991
Subject: Analysis of Standard Data Formats

I was informed by Bob Seals that you would be interested in the LaRC DAAC's analysis of standard data formats. We evaluated netCDF and HDF. Enclosed is a copy of the requirements and report outline provided by Ted Meyers, VO Data Formats Task Leader, and a copy of the LaRC DAAC's report evaluating these two formats. If I can answer any questions for you, please call me at (804) 865-7909 or send e-mail at travers@sabre.larc.nasa.gov.
Section 3 - General SDF Activity

The questions in this section pertain to your overall SDF activities. Answer these questions once.

3.1 Comparison

Compare data formatting systems. If you used more than one data formatting system then compare those systems. Otherwise compare the data set you created with the input data set you started with. If you put the same data set into two systems use that for comparison. If this was a new data set then try to provide a subjective comparison with your experience with another formatting system that you commonly use.

3.1.1 Compare the size (number of bytes of storage media) of the data set in various formats.

3.1.2 Compare the speed of ingesting the data set in various formats.

3.1.3 Compare the ease of accessibility of the data. (e.g., can it be read with a text editor or simple user written programs?)

3.1.4 Compare the length of time it would take a user to learn how to ingest a data set in various formats:
   a) If this was the first time they had used this formatting system.
   b) If they were familiar with the formatting system but not with this particular product.

3.2 Recommendations/suggestions

Include here any current recommendations about:

3.2.1 Adoption or deletion of any of the formatting systems under study.

3.2.2 Modifications that should be made to a given formatting system.

3.2.3 Additional tests that should be done on a given formatting system.

3.2.4 Data sets at your DAAC or location that should be put into a given formatting system (either in support of science use or to test other features of a formatting system).

3.2.5 Anything else you care to recommend on this topic.
Enclosure 2 - Compliance of the Data Format With Version 0.3 Requirements

Report how the system(s) you studied meet each of these Version 0.3 Requirements. Score on the basis of:

1. Fully meets the requirement.
2. Mostly meets the requirement. Describe what problems exist that prevent this from being scored 1. What work arounds or alternatives exist that would allow this to be scored 1.
3. Somewhat meets the requirement, no major problems. Describe the problems in a narrative note for any item rated 3.
4. Significant problems with this requirement but a work around is possible. Describe the problems and the required work around in a narrative comment for any item rated 4.
5. Fails to meet the requirement. No reasonable work around exists. Describe the problem(s) and explain why no reasonable work around exists.

NC If you did not evaluate a feature or do not know the answer respond NC and explain why.

Note: The Version 0.3 Requirements are currently under review. They are presented here as they were distributed for review. Any additional comments on the requirements themselves would also be appreciated.

Table for Comparing Compliance of the Data Format With Version 0.3 Requirements

<table>
<thead>
<tr>
<th>Data Format Requirements</th>
<th>Score</th>
<th>Explanation/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Hardware / Operating System / Language / Media/ Comm / Independence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1. The format system must support automated exchange of data in a heterogeneous data processing environment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2. The format system must be independent of unique or vendor specific hardware, storage media, networks, communications protocols, operating systems, compilers or other special software except for software interfaces and utilities provided explicitly as part of the format system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3. The format system must provide data access from as many types of computers as possible; the interface software and utilities must be as portable as practical and run, at a minimum, on the operating systems most commonly used in the scientific community.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Standard Data Formatting System Evaluation Report  
LaRC DAAC  
October 1991

Section 1 - Overall Description of Effort

1.1 The LaRC DAAC studied the Hierarchial Data Format (HDF) and the Network Common Data Form (netCDF) formatting systems. The sample data set was the 32-bit scanner data from the ERBE S-4G product. There has been a minimum number of problems during this effort. Most problems were caused by an unfamiliarity with the formats and in understanding the documentation, particularly with netCDF. The results of our effort thus far have been to successfully convert the sample data set to both HDF and netCDF and validate the converted data. In addition, a Fortran program that originally browsed (in tabular form) the S-4G data was modified to browse the data from a native-format file, an HDF file, or a netCDF file. This modification was easily accomplished by adding subroutines to make calls to the HDF or netCDF libraries.

1.2 The LaRC DAAC proposes to continue the study of standard data formats as follows

1. Evaluate NCDS's Common Data Format (CDF) Version 2.1  
2. Create new and useful data sets in HDF and CDF formats  
3. Distribute HDF and CDF formatted files to our science community for evaluation.  
4. Determine the standard format most suitable for the initial implementation (FY92) of the LaRC DADS.  
5. Evaluate the level of DAAC support required to introduce a new formatting system to our science community.  
6. Assist the V0 Project in other activities associated with the selection of a standard formatting system for EOS data.

Section 2 - Individual SDF Activity - HDF

2.1 Standard Data Format Description

2.1.1 Hierarchial Data Format (HDF)  
2.1.2 Have not used HDF before this study.  
2.1.3 We are not aware of any users of HDF in our local science community.  
2.1.4 The software and documentation for HDF is freely available.  
2.1.5 Yes, the documentation adequately describes the system.
For this example, 32-bit scanner data is used. This means that the packing efficiency in "common formats" will be good.

Gives user learning time to ingest data

Section 3 - General SDF Activity

3.1 Comparison

3.1.1 Native format - 3652608 bytes includes scale factors and spares.
HDF - 364019 bytes floating point; no spares.
netCDF - 3617800 bytes floating point; no spares.

3.1.2 We compared the ingest time between all three formats
native format - 24.6 cpu seconds
HDF - 23.4 cpu seconds
netCDF - 15.0 cpu seconds

3.1.3 Both HDF and netCDF files can be easily accessed with a simple user written program.

3.1.4 a) The length of time to learn how to ingest a data set if they had not used this formatting system but were familiar with the data set.
HDF - 16 to 24 man hours
netCDF - 64 to 80 man hours

b) The length of time to ingest a data set if they were familiar with the formatting system but not the data set.
HDF - 56 to 80 man hours
netCDF - 56 to 80 man hours

3.2 Recommendations/Suggestions

Because of problems with netCDF we prefer not to evaluate it any further but suggest that we continue our study of HDF and begin a study of CDF.

* Dave Fuller (netCDF) can't understand why it took so long.
SOFTWARE FOR PORTABLE SCIENTIFIC DATA MANAGEMENT

Stewart A. Brown, Mike Folk, Gregory Goucher, & Russ Rew

The problem of processing large amounts of scientific data is a multifaceted one, and many different groups are addressing parts of the problem. Observational data are pouring in from various data-gathering networks and instruments. Numerical simulations generate vast amounts of data.

As the processing power of computer systems increases, the problems of getting large volumes of data into and out of applications and of visualizing the results have become significant obstacles to gaining insight from the data. A related problem is how to organize data sets with a view to the various kinds of access that applications demand.

Often a diverse range of computer systems is used for processing related data. A numerical simulation may be set up on one workstation, run on a large parallel computer, and the results visualized on a different graphics workstation. Some or all of these computers may have different binary data representations. This presents its own set of problems.

In this article we discuss the general philosophy of portable scientific-data-management systems and compare four of the systems that address these kinds of problems: CDF, HDF, netCDF, and PDB.

Introduction

Traditional methods for handling scientific data include the use of flat sequential files of machine-specific binary data or portable ASCII data. For applications such as visualization, such methods are inefficient in storage, access, and ease-of-use. Relational database systems fail to accommodate the multidimensional or hierarchical structures often found in scientific data sets. In addition, relational systems do not provide adequate performance for the size, complexity, and type of access required for many scientific data sets. Object-oriented database systems may provide solutions to some of these problems in the future, but relevant experience with such systems is still meager.

In the mid 1980s, efforts at several institutions identified and attacked these problems using various approaches. The systems that were developed independently have features in common, as well as differences that reflect the emphases and requirements among the institutions involved. Three of the systems (CDF, HDF, and PDB) developed from independent models of scientific data; a fourth (netCDF) was built on the CDF data model with an independent implementation designed to support different trade-offs. Some consolidation of features is now occurring; for example HDF now supports the CDF/netCDF data model using the HDF format.

These systems address the need for portability of both data and software, for flexible control of the organization and contents of data files, and for ways to make data self-contained and self-describing. If a data set is self-describing, general-purpose tools can extract not only data values, but also information about the data values (e.g., units) and relationships among components of the data (e.g., whether a set of values represents coordinates for other data).

The systems that we discuss here are implemented as software libraries. This provides a great deal of flexibility for users: applications can be built using a library's data-access interface to deal with data at a higher level than bits and bytes. Each system also defines an associated format for files that are accessed by calling routines in the associated library.

We wish to emphasize two meanings of portability when discussing these systems. The first meaning refers to the data; the second, to the libraries. Data may be stored in a file using a portable data representation. Such a file can be moved among computers with different architectures, and the data in the file can be accessed without an explicit conversion step. The various libraries achieve this data portability in different ways, but all can handle data format conversions implicitly within the library. Second, the libraries are implemented in a relatively portable way, so that application programs that use a library for data access can be moved from ma-
Table I. Comparison of four scientific-data-management systems.

<table>
<thead>
<tr>
<th>Feature</th>
<th>CDF</th>
<th>HDF</th>
<th>netCDF</th>
<th>PDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages supported</td>
<td>C, Fortran</td>
<td>C, Fortran</td>
<td>C, Fortran, C++</td>
<td>C, Fortran, SCHEME</td>
</tr>
<tr>
<td>Inherent data types</td>
<td>char, short, int, float, double, string.</td>
<td>byte, short, long, float, double.</td>
<td>byte, char, short, long, float, double</td>
<td>char, short, int, long, float, double, pointer</td>
</tr>
<tr>
<td>User-definable types</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Data-conversion method</td>
<td>XDR, native</td>
<td>XDR, native</td>
<td>XDR</td>
<td>PDC</td>
</tr>
<tr>
<td>Maximum array dimensions</td>
<td>10</td>
<td>unlimited</td>
<td>32</td>
<td>unlimited</td>
</tr>
<tr>
<td>Extended array dimension</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Hyperset access</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>User-definable attributes</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>Attribute types</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>Named dimensions</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Array-index ordering</td>
<td>row, column</td>
<td>row, column</td>
<td>row, column</td>
<td>row, column</td>
</tr>
<tr>
<td>Shareability</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Compression</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Supporting tools</td>
<td>yes</td>
<td>many</td>
<td>ndump, ngen, a few others</td>
<td>PDBView, PDBDiff, ULTRA II</td>
</tr>
</tbody>
</table>

Machine to machine by merely recompiling the programs.

To facilitate a comparison among the four systems (see Table I), we use the following list of features:

- Languages supported. The library can be used from these languages.
- Inherent data types supported. This refers to data types (such as integer or double) for which the library has built-in support.
- User-definable data type supported. Does the library permit applications to define and use their own data types (other than arrays, which are supported by all the libraries)?
- Data-conversion method. How is data portability achieved?
- Maximum number of dimensions. This applies to an array managed by the library.
- Extendible array dimension supported. Does the library support at least one dimension that may be extended as data are added to a data set?
- Hyperset access. Does the library support access to subsets of a larger data set, such as cross-sections?
- User-definable attributes. Does the library support a fixed set of attributes that may be attached to the data (e.g., "units"), or are user- and application-definable attributes supported?
- Attribute types. What data types does the implementation allow for attribute information?
- Named dimensions. Does the implementation support the concept of named dimensions?
- Array index ordering. Does the implementation support row-major or column-major array-index ordering (or any other)?
- Shareability. Does the implementation allow one writer and multiple readers to access the same data concurrently?
- Compression. Does the implementation support data compression?
- Supporting tools. Are there tools for browsing files, visualization applications, and data analysis?

For brevity, we have not included some features that are common to all four systems. For example, all the systems support some backward compatibility with older versions of the associated format as the system evolves. Also, all four systems are freely available; information about how to obtain the software appears in the box labeled I/O-Paul Dubois on p. 308.

CDF

The Common Data Format (CDF) software is a scientific data-management package that was designed and developed at the National Science Data Center (NSDCC) at NASA's Goddard Space Flight Center (GSFC). The development of CDF arose out of the recognition by the NSDCC of the need for a class of data models that is matched both to the structure of scientific data and to how such data may be used. Even though CDF has its own internal self-describing format, it is more than just a data format. It is a library that allows programmers to access and manage multidimensional data in a fashion consistent with its scientific orientation. The irony of the term "Format" in the name of this soft-

COMPUTERS IN PHYSICS, VOL. 7, NO. 3, MAY/JUN 1993 305
ware is that the actual data format utilized by CDF is of no concern to the user, since it is only accessible through the 23 interface routines that insulate the user from needing to know anything about the format.

The CDF library was designed to provide the essential framework from which generic applications (e.g., visualization, statistical analysis, browsers, etc.) can easily be created. The library allows developers to create applications that permit users to slice data across multidimensional subspaces, access entire structures of data, perform subsampling of data, and access one element independently of its relationship to other elements.

The concept of using an internal data dictionary to describe the contents of a data file is not new for the purpose of achieving a data-independent transportable standard. However, the CDF differs from those earlier formats by being oriented toward the researcher’s view of the data. The most important difference between the CDF and conventional data formats is in the self-describing nature of the data descriptions maintained within the CDF and its supporting software. This self-describing property makes it possible to use CDF for data from a wide variety of disciplines.

The CDF library supports two storage models: a multiple-file model, in which one file contains all the metadata and there is one file for each variable, and a single file model, in which all the data reside in a single file. Each storage model has advantages, depending on how the data will be managed, accessed, or updated.

When generating a CDF file, the programmer may choose either of two encoding schemes: native encoding, in which a machine’s native binary representations are used, and network encoding, in which the data are transparently converted from the native format to a standard data format when writing the data and from the standard format to native format when reading the data.

HDF

Hierarchical Data Format (HDF) was created at NCSA to provide users with a file format for sharing scientific data in a heterogeneous computing environment, to provide a set of high-level interfaces to that data, and to support the development of scientific visualization and analysis tools that have a high degree of data-set independence.

HDF provides simple primitive objects out of which more complex objects can be built. Each type of primitive object is identified by a “tag.” The basic structure of HDF consists of an index with the tags of the objects in the file, pointers to the data associated with the tags, and the data themselves.

The design of HDF reflects the assumption that we cannot anticipate what types of data objects will be needed in the future, nor can we know how scientists will want to view their data. As new science is done, new types of data objects are needed, and new tags must be created. The HDF library contains programming interfaces designed to provide views of the data that are most natural for users. As we learn more about the way scientists need to view their data, we can add new interfaces that reflect data models consistent with those views.

HDF supports most common types of data and metadata that scientists use, including multidimensional arrays, raster images, polygonal mesh data, tables, and text. In the future there will probably be a need to incorporate new types of data, such as voice and video.

The HDF library currently supports the following application programming interfaces (APIs) and their corresponding data objects:

- A “general purpose” API for doing basic I/O operations.
- “Raster image set” APIs for accessing raster images.
- A “scientific data set” (SDS) API for accessing multidimensional arrays of the primitive types.
- An “annotations” API for reading and writing textual annotations.
- A “vdata” API for accessing sequences of records in which the fields consist of different primitive types.
- A “Vgroup” API for accessing and managing groups of objects.

There is much overlap between the APIs supported by HDF and those of the netCDF, CDF, and PDB formats. With respect to netCDF this overlap will be total when HDF version 3.3 is released. A project supported by the National Science Foundation will result in the implementation of the netCDF data model within HDF.

netCDF

The netCDF interface was designed to support the creation, access, and sharing of data that are portable, self-describing, directly accessible, and appendable. “Directly accessible” means that a small subset of a large dataset may be accessed efficiently, without first reading through the preceding data. “Appendable” means that new data can be appended efficiently to an existing netCDF file. The netCDF data model evolved from an early version of the NSDC CDF data model, adding aggregate and hyperslab access, named dimensions, variable-specific attributes, and conventions for coordinate variables. The netCDF implementation first featured the use of the XDR standard for portable data representation, a single-file implementation, and an ASCII-based language for representing the binary data in a human-readable and editable form.

NetCDF was initially developed to be used within the Unidata Program as an interface between system-level programs that capture broadcast meteorological data and application programs that analyze and display the data. Many other groups have since found netCDF to be useful for sharing data among different architectures, providing a flexible way to access data cross-sections, or providing an extensible interface to data that insulates application programs from the details of a data format. NetCDF emphasizes a single common interface to data, implemented on top of a platform-independent representation.

To achieve data portability, netCDF relies on the XDR standard for external data representation. Use of a
vendor-supplied XDR library included with most UNIX systems is recommended, but a portable implementation of XDR (made freely available by Sun Microsystems) comes with the netCDF software distribution for use with operating systems that do not include an XDR library.

For more information about what platforms are supported, what visualization tools can import netCDF data, what utilities are available, how to subscribe to the netCDF mailing list, and answers to other frequently asked questions, use anonymous FTP to get the file pub/netcdf/FAQ from host unidata.ucar.edu.

PDB

PDB began with an attempt to provide the ability to save and restore the kinds of data and structures used in C programs. In fact, the basic form of the API (independent of language bindings) was intended to be as similar as possible to the standard C I/O library interface. The hope was and is to make it as easy as possible for C programmers to save their data structures easily and not have either to: avoid using natural data structures for fear of I/O difficulties; or write endless special-purpose I/O routines for each data structure defined. By meeting this goal, support for Fortran, which has more limited data-structuring capabilities, was easy. Functionality aimed primarily at Fortran usages would not provide the needed flexibility to C programs.

PDB is implemented in C and one component of the Portable Application Code Toolkit (PACT) set of tools. PDB relies on no software outside of PACT, but it does derive its portability from other parts of PACT.

Differences

Although they started from almost the same data model, which was developed at NSIDC for an early version of CDF, netCDF and CDF have evolved independently and use different file formats. Other differences are that netCDF does not support native-mode representation or multiple datasets for the efficient addition of new variables, and CDF does not support named dimensions or simple conventions for variables coordinates.

NetCDF data, CDF network-encoded data, and some kinds of HDF data are machine-independent: the form in which such data is stored is the same for any platform on which the library is implemented. PDB data, CDF native-encoded data, and other kinds of HDF data use the native representations for the machines on which the data are first written, converting the data only when necessary to native encodings of other machines. Each of these two approaches has advantages. Machine-independent data can be shared transparently across network file systems. With machine-independent data, conversion occurs whenever data are accessed, and for some kinds of data (e.g., byte arrays or floating-point numbers on machines that support IEEE floating-point representa-
SCIENTIFIC PROGRAMMING

...tions), the necessary conversions are trivial. On the other hand, native representation incurs no conversion cost for any kind of data when accessed on the machine for which the data were written.

PDB by default writes data to files in the native format of the machine on which the software is running. It keeps a parametrization of the binary formats in which the data are written in the file, so that when the file is read, conversions can be done to a different format. PDB also permits the file to be written in the binary format of some target system. In this way, application developers can tailor their systems to the expected usage. For example, data generated on a Cray can be written to a file aimed at a slower PC, so that when the file is accessed on the target PC, no conversions need to be done, and access speed is maximized.

HDF uses a centralized registry of basic data tags. The HDF tags designate fundamental data types needed in application programs, but because these tags are open-ended, unanticipated data types may be added. HDF can identify complex data structures with a small tag, and so it can store such structures compactly. PDB can store arbitrary C structures, including complex user-defined data structures. The other interfaces have somewhat less flexibility, supporting arrays, structures for which a tag has been defined, or simple records composed of a sequence of scalars and arrays of various types. CDF, netCDF, and PDB are somewhat more self-describing than HDF, since it is not necessary to agree upon or register a data-structure tag.

HDF is a single format that supports a number of interfaces; hence there are a large number of routines with simple interfaces in the HDF library. Users generally use very few of these routines. For example, only two HDF routines are generally used for reading and writing raster images. In contrast, CDF, netCDF, and PDB are single interfaces for many different kinds of data. One of the aims of these interfaces is to have a small "surface area," so that the entire interface is easy to learn and use. Hence there are a smaller number of more-general-purpose functions in the CDF, netCDF, and PDB interfaces. In either case, a small fraction of the routines in the interface fulfill a large proportion of the most common needs.

We know of no all-purpose conversion packages that will handle data in all of these formats, but at least two commercial visualization packages can import data from three of the four formats. It is not always possible to represent all the information in a file of one format in one of the other formats, since the systems were designed for somewhat different purposes, and may use different conventions for representing some data relationships. But we believe that all of these packages have advantages over traditional methods for storing and accessing scientific data.

Further reading

CDF

HDF
NCSA HDF Specification Manual. (Available from NCSA or by anonymous FTP from ftp.ncsa.uiuc.edu.)
NCSA HDF Calling Interfaces and Utilities. (Available from NCSA or by anonymous FTP from ftp.ncsa.uiuc.edu.)
netCDF
PDB

I/O—Paul Dubois

Due to production schedules I have not yet begun to receive your comments, ideas for articles, or information to supplement that given in previous columns, but this is a good place to remind you that all of these are eagerly sought. One of my fellow editors commented to me that editing a column like this is like selling insurance: once you run out of friends and relatives to ask for articles, it gets tough. If you cannot e-mail me at dubois1@llnl.gov, please write to me at L-472, Lawrence Livermore National Laboratory, Livermore, CA 94550, USA.

Here is how to get the portable database software mentioned in this month's article:

(1) The CDF V2.3 distribution is available via anonymous FTP and NSInet for VMS systems, via anonymous FTP only for UNIX systems, and via anonymous FTP and floppy disks for MS-DOS systems.

There is a user support office (USO) for CDF that you should contact when you need the CDF software, programming help, or assistance for any other form:

Phone (301) 286-9506
Internet—CDFSUPPOR T@NSSDCA.GSFC.NASA.GOV (128.183.36.23)
NSInet (SPAN)—NSSDCA::CDFSUPPOR T.

(2) HDF is available on Internet via anonymous FTP (ftp.ncsa.uiuc.edu). Look in directory HDF for latest version of the library. Look at top level for various tools.

A mailing list known as hdfnews exists for discussion of the HDF interfaces and announcements about HDF bugs, fixes, and enhancements. To subscribe, send a request to sxu@ncsa.uiuc.edu. Information about HDF documentation can be obtained from the same address.

(3) netCDF is available on Internet via anonymous FTP (host unidata.ucar.edu, file pub/netcdf/netcdf.tar.Z)

(4) PDB is available on Internet via anonymous FTP (host phoenix.oof.llnl.gov, file pub/pactxx_xx_xx.tar.Z)
Section 8

Data Provided by NCAR Data Support
Data Provided by NCAR Data Support

We will discuss some of the aspects of data help provided by the NCAR Data Support Section.

A. Impact of Changing Formats on Users

Many PIs have obtained datasets from NCAR over the last 25 years. Each year we send about 400 copies of datasets around the world from NCAR. A similar number are used online at NCAR. When data are sent to other countries and laboratories, they often let several other PIs use it or send copies to other people. The average rate of secondary use is probably 4 or 5 PIs. In addition, a given PI often uses the same data for several projects. At least one country used NCAR data to set up a data bank that many universities can access. Some NCAR datasets have been put on CD-ROMs, purchased by about 200 people. We, therefore, see that a lot of people are using data in the archive formats.

When people obtain an update to a given type of data, they prefer to receive the data in the same format they got before. This is the "standard format" to them. About 1986, we felt that it was time to steer people away from the NMC octagonal grid format and encourage them to use the full hemisphere equivalent (or the 2.5\(\text{\textdegree}\)delat-long grid). We soon found that there was so much software connected to the former format that it made better sense to continue it as one choice (and probably it is still the most used choice).

B. Some Attributes of Desirable Data Structures

Fixed-length blocking structures can be handled by almost all computer operating systems. However, for many data problems we need to be able to handle variable length information. For example, some balloon reports of upper air data have 10 levels, others have 80. Also, we may need to append some error information to only one report in 1000. Then, the report becomes longer.

The data often needs to be in fixed length packets for communications, and in fixed buffers (where the size can be set) for residence on disks or tape storage media.

The general idea with data structures is to have enough standards in how data are blocked and routed and stored that people do not have to worry about the details of how this happens. These functions can (and should) all be carried out without any knowledge about the content and format of the data.

Please refer to Jenne, 1989 (7pp) for more information about data structures.


General aspects of data formats: See Apx 2 for a discussion of what to look for in formats. What attributes does a good format have?

C. What Type of Data Access Do Users Need?

Our main concern at NCAR is to help PIs obtain our data in Fortran arrays, ready for calculations, and with almost no learning time. We usually send the data in the archive
format. Also, we try to provide portable software to unpack the data into a Fortran array, when desired by the user. This software will work on nearly all target computers.

This procedure is very fast, it considers the Fortran array the main standard across computers, and it permits the user to manipulate the data in any way that he or she wants to. These methods efficiently support large calculations, and they also support browse of smaller amounts of data.

Suppose that a PI wants to use the manipulation capability of an image processor, a spreadsheet, a statistics routine, a canned graphics routine, or the NASA climate displays or Unidata. To facilitate one of these choices, it should be rather easy to import the Fortran array into the main software programs that are popular.

NCAR DSS priorities: We try to insure that the support for calculations and user-browse work smoothly, and will offer limited data access help for people to write the translators needed for insertion of the data into spreadsheets, etc. The data source can be on-line at NCAR or delivered by tape, CDROM, cheaper tape, or com.

D. Data Transportability Across Systems

NCAR concentrates on methods that permit a high level of data transportability, while still permitting efficient data packing.

If data are stored in a character format, a Fortran statement that will unpack the data on one type of computer will also work on another. It is not practical to save high volume data in a character format (data such as much satellite data and analyses). Transportability for routines to unpack the packed binary data has been achieved by using GBYTES (get bytes) routines defined by NCAR in 1965.

It is not quite as simple as this because of different data blocking structures (and block length limits) in various systems, and because of byte swapping in VAX formats. In general, it is still simple if data are in fixed length block structures. Other structures can still be handled if GBYTES is available.

In general, these methods permit us to achieve several goals: (1) transportability of unpacking software, (2) ability to handle a diversity of formats and still easily transport data and (3) permit binary packing (compression) of data, with ease of transporting and unpacking.

The remaining national problem is to achieve fewer variations in data blocking structures (how individual logical records are arranged into larger blocks).

E. User Program Codes

To accomplish major calculations that need data, the data input methods should be simple, and be easy to understand. It is best to keep the user codes free of special calls that make them less portable, and more tied to any one data scheme. The thing that is likely to remain most standard is the Fortran array.
Attachment I
Data Packaging and Aspects of Good Formats

A. Some General Aspects of Data Formats

1. Data Packaging

Now most computer systems can handle fixed length blocking of reports. It would be nice to be able to handle variable length blocking in a more general way. Checksum protection should be included in the archive.

2. Data Transfer and Storage

The slowest links in computing systems are communications and the links to external storage. If a given format increases the data volume by a factor of two, this equivalent to cutting the channel speeds in half, and doubling the amount of data that must be stored. Other tradeoffs are also involved, so that in some cases, we should accept the volume penalty, but it must be thought through.

B. Aspects of Good Formats

We prefer to give local groups a reasonable amount of latitude in defining a needed format, but there are some features that we look for to help determine the overall quality of the data structures:

1. Is it possible to fairly readily sort the reports in the structure that they are in?

2. If we have one file of "oranges" and another of "apples", is it fairly simple to merge the two if necessary? This usually requires variable length capability in the blocking structures.

3. Does the overall structure permit variable length reports?

4. If we need to attach something to one report in 100, this should be possible (it gives variable length data). Bufr code supports this.

5. It is necessary to have fast ways to handle the data, and to ignore unneeded data. Considering this, the blocking structures should include length of reports, and the ends of reports should usually be on 64-bit boundaries and not on the boundary of a random number of bytes.

6. If a type of data includes 20 variables reported 95% of the time and 30 reported only 1%, it is useful to use fixed slots (for quick access) for the 30 common variables and a different method for the others (gives variable length data).

7. It should be easy to select a subset of the data, and re-block the output data for delivery to the user.
Attachment II
Formats for a Big Project

We will consider the data format decisions for a large project, and will use the large project to prepare the world's marine ship data as an example (project is called COADS). Under this project ship and buoy data from 1854-present are prepared. Three main laboratories cooperate to make this project happen: NCAR and ERL/Cires in Boulder, CO and NCDC in Asheville, NC. In addition, MEDS in Canada has helped to clean up the world's drifting buoy data. The US National Buoy Center has prepared fixed buoys and many other centers have also helped.

During 1983-84, the world's data for 1854-1979 was prepared. The overall problem was to input observations from 15 different data streams, clean up problems, eliminate duplicates, etc. Participants from NCAR and Cires in Boulder defined an efficient binary format that compressed the volume and permitted the 50 Gigabits of input data (with 100 million reports) to be readily sorted. In the first stages, we kept the format somewhat fluid so that needed flags could be added as the need was determined. Later on, we found that some other information needed to be carried along for some reports. It was appended at the end of each report.

We needed this format flexibility to achieve the scientific goals of the project. We needed efficiency to do it at reasonable cost. The data had to be input and known problems fixed. It had to be sorted and merged. Then quality control and duplicate elimination programs were run. Finally the data was condensed and resorted for input to statistics programs.

The resulting observed data was returned to the data center (NCDC) at Asheville. They needed it in a character format so that they could easily send it to many less sophisticated users. Also there are Federal standards that mandate that they use characters (in ASC II form) unless special permission is given. Therefore the Boulder crew agreed to convert the data to a character format. NCDC had defined a character format. The Boulder crew felt that other variables (needing 4 characters) would be important for some users. NCDC didn't want to change the format again (which had already changed a few times in the past years) so Boulder just provided the desired form. Since the two variables (and other information not sent to) are still in the basic format, this decision didn't matter much.

In this example, we see that the details of the format need to involve those very familiar with the data. For large datasets, efficiency also becomes very important! Ease of programming and automatic control to guard against data loss (or unwanted changes) is vital. If several high level committees had been involved in these format decisions, we believe that the decisions would have been worse, the process would have taken forever, and it would have cost a lot.

Note that some of the input data streams held data from many countries, where the content and format had been determined by international committees that knew this data. These committee efforts were helpful and appropriate. We couldn't just adopt one of those formats because we had to include appropriate source and control information. We also needed to limit the volume.
Section 9

Two Items Toward an Analysis of Formats
Data Handling and Distribution by NCAR DSS

This text was prepared because of various questions about the preparation of datasets, desirable data structures, how to make data easy to use, and how the problems of data transportability across different computers can be solved. Much of our present data will need to be easily used 50 years from now. The associated format requirements are briefly discussed. There is also a section about the future handling of logical records. This introduction promises a little too much, but most of these procedures have been working well for some time.

The NCAR Data Support Section prepares many large datasets. We also receive many datasets from around the world. In some cases we need to change data formats. In most cases, we can adapt our procedures to just accept the format that is provided. This reduces the hassle on data providers. We will now discuss the procedures for processing large datasets at NCAR.

1. Process Large Sets of Observed Data

In Data Support, a considerable amount of our time goes to preparing very large datasets that have individual observations, often with global coverage and covering many years of time. The requirements for handling these large projects are:

- Very compact formats, but usually still simple
- Variable length capability
- Checksum control so there are no mechanical errors
- Absolute volume control (so that a human or hardware error can’t lose data).
- Data flow methods that permit very fast data paths and fast sorts (that use the 7-way sort-merge methods).
- Control of the format so that all needed variables and flags can be included. In this process, we cannot afford the time to coordinate with 100 other groups. It takes a lot of coordination as it is, and the results would be hurt by letting all of the world’s format questions get mixed up with the science.

What are some of these projects?

- COADS project: prepare the world’s ship and buoy data, from 1854 - on
- World aircraft data: prepare data for 1960 - on for reanalysis
- World rawinsondes. Prepare data for 1948 - on. This includes data from GTS (telecom in real time), as well as data from the archives of many countries
- Satellite cloud-drift winds. Prepare a dataset
Run control automation

The DSS has developed methods of run control automation so that we can accomplish large data development tasks with limited staff. The methods automate the submission of thousands of jobs, tell the program where to find data inputs, and keep absolute control over report counts and data volume. Thus, no data can be lost without our knowing it. All the data are under checksum control. Examples are methods used for handling the data flow for the COADS project (collect all world ship reports), and methods used to save 12 Tbits of satellite data.

When we started the COADS project in 1982, we figured that more than 5,000 job submissions would be required. There is no way to do this manually and avoid human errors, and systematically check for machine errors. Therefore, these methods for automating the tasks were developed.

Common formats

Where do common formats like Net-CDF fit into these projects? This format doesn’t have the properties that are needed for projects like these. A conversion to Net-CDF could be made later, but it would blow up the volume a lot., The people who just want to use the data would be better off with the basic format (with access routines), rather than a format like Net CDF.

2. How are Datasets Used?

Data are used in several different ways. The categories of use of data are:

1. Large batch computing jobs
2. Personal browse
3. System browse

Data Support (DSS) has concentrated on doing a good job with No.1, a reasonable job with No.2, and we make the data available for other groups to support No.3.

University of Wisconsin, University of Maryland, Unidata and others concentrate on doing No.3.

I have been in several meetings where scientists say: “please give us the data so that we can do our own thing.” This is often interpreted as meaning bigger and better display systems. The PI really means that he/she wants easy access to the data to use for calculations, or to use in display routines that the PI chooses.

It often amuses me that people use the high data rates of EOS to justify big data systems, and then design data systems that are not appropriate for handling large volumes of data. One of our problems in national data activities is that some groups focus so strongly on item No. 3 that they ignore the need for 1 and 2, or try to force their methods on 1 and 2 so that these tasks cannot be done as easily or efficiently.

It is still true that there is a real need for graphics of many types. Some of this capability will be developed within our scientific community. Some will be commercial software. The user should be able to match up the data with software that he/she chooses.
3. Steps to Use Data

What are the procedures used to access data? The methods used to process data rapidly and easily are as follows:

- Move a logical report to a buffer

  One logical report is taken from its blocking structure and put into a program buffer, ready to be unpacked. This logical record might be a surface synoptic report, a rawinsonde observation, a monthly record having daily maximum/minimum temperatures and precipitation, a gridded analysis, or a satellite scan line.

- Unpack the report in the buffer

  Now that the report is in the buffer array, it is easy to unpack it. If it is in a character format, ordinary decode statements can be used to prepare numbers that are ready for calculations. If it is in a packed binary format, the Gbytes routine can be used to extract the packed numbers and prepare the usual full-word numbers that are ready for calculations. The user ordinarily does not have to worry about the details of this unpacking process. An access routine is supplied that accomplishes the tasks. But the user does some programming to call the access routines. The Gbytes routine is briefly described in Attachment 1.

  Other organizations also use these methods to distribute large amounts of data, and make it easy for users to access the data. For example, ECMWF sends out large amounts of gridded analyses. They provide access routines that use the Gbyte subroutines obtained from NCAR.

  These methods make it possible to process large amounts of data very rapidly. They are easy to learn, and low in cost to apply. They are efficient in the use of computer time and keep the data flow fast.

4. User Program Codes

  To accomplish major calculations that need data, the data input methods should be simple, and also they should be easy to understand. It is best to keep the user codes free of special calls that make them less portable, and more tied to any one data scheme. The thing that is likely to remain most standard is the Fortran array of numbers.

5. Impact of Changing Formats on Users

  Many PIs have obtained datasets from NCAR over the last 25 years. Each year we send about 400 copies of datasets around the world from NCAR. A similar number are used online at NCAR. When data are sent to other countries and laboratories, they often let several other PIs use it or send copies to other people. The average rate of secondary use is probably 4 or 5 PIs. In addition, a given PI often uses the same data for several projects. At least one country used NCAR data to set up a data bank that many universities can access. Some NCAR datasets have been put on CD-ROMs, purchased by about 200 people. We, therefore, see that a lot of people are using data in the archive formats.

  When people obtain an update to a given type of data, they prefer to receive the data in the same format they got before. This is the "standard format" to them. About 1986, we
felt that it was time to steer people away from the NMC octagonal grid format and encourage them to use the full hemisphere equivalent (or the 2.5° lat-long grid). We soon found that there was so much user software connected to the former format that it made better sense to continue it as one choice (and probably it is still the most used choice).

6. Some Attributes of Desirable Data Structures

Fixed-length blocking structures can be handled by almost all computer operating systems. However, for many data problems we need to be able to handle variable length information. For example, some balloon reports of upper air data have 10 levels, others have 80. Also, we may need to append some error information to only one report in 1000. Then, that particular report becomes longer.

The data often needs to be in fixed length packets for communications, and in fixed buffers (where the size can be set) for residence on disks or tape storage media.

The general idea with data structures is to have enough standards for how data are blocked and routed and stored that people do not have to worry about the details of how this happens. These functions can (and should) all be carried out without any knowledge about the content and format of the data.

Please refer to Jenne, 1989 (7pp) for more information about data structures:


General aspects of data formats: See Appendix 2 of another text for a discussion of what to look for in formats. What attributes does a good format have?

7. Data Access for Scientists

There are some scientists who are primarily interested in the system’s aspects of computing and data, but I believe that up to 95 percent of working PIs would agree with the following:

• We want easy access to the data with low learning times. The low learning time needs emphasis.

• In the data “shop” we want to be able to talk with someone who knows something about the mechanical and scientific attributes of the data.

• We want to use the data for calculations and in display systems that PIs choose, not that are dictated by the data system. Some system browse is reasonable, but it does not give more than 10 percent of what PIs need.

• A key requirement of good computing working environments for scientists is that it must be easy to obtain data, and it must be easy to insert data into various software tools. Many of the latter tools will be from commercial sources.

• Scientists know that there are formats that are trivial to learn to ingest the data they need. They fear that they will be forced to use something much more complicated and cumbersome in order to obtain any data.
• It should be quite easy to ingest data, even if a PI only uses the procedure once each six months.

• The PI will choose software on the basis of usefulness, ease of use, and cost.

• There is a tendency for people to design systems that are costly, inflexible, and hard to use, with the explanation that the users requested it.

8. What Type of Data Access Do Users Need?

Our main concern at NCAR is to help PIs obtain our data in Fortran arrays, ready for calculations, and with almost no learning time. We usually send the data in the archive format. Also, we try to provide portable software to unpack the data into a Fortran array, when desired by the user. This software will work on nearly all target computers.

This procedure is very fast, it considers the Fortran array the main standard across computers, and it permits the user to manipulate the data in any way that he or she wants to. These methods efficiently support large calculations, and they also support browse of smaller amounts of data.

Suppose that a PI wants to use the manipulation capability of an image processor, a spreadsheet, a statistics routine, a canned graphics routine, the NASA climate displays or Unidata. To facilitate one of these choices, it should be rather easy to import the Fortran array into the main software programs that are popular.

NCAR DSS priorities: We try to insure that the support for calculations and user-browse work smoothly, and will offer limited data access help for people to write the translators needed for insertion of the data into spreadsheets, etc. The data source can be on-line at NCAR or delivered by ordinary tape, CDROM, cheaper tape, or communications.

9. Data Transportability Across Systems

NCAR concentrates on methods that permit a high level of data transportability, while still permitting efficient data packing, and ease of use.

If data are stored in a character format, a Fortran statement that will unpack the data on one type of computer will also work on another computer. It is not practical to save high volume data in a character format (data such as much satellite data and analyses). Transportability for routines to unpack the packed binary data has been achieved by using GBYTES (get bytes) routines defined by NCAR in 1965.

It is not quite as simple as this because of different data blocking structures (and block length limits) in various systems, and because of byte swapping in VAX formats. In general, it is still simple if data are in fixed length block structures. Other structures can still be handled easily if GBYTES is available.

In general, these methods permit us to achieve several goals: (1) transportability of unpacking software, (2) ability to handle a diversity of formats and still easily transport data and (3) permit binary packing (compression) of data, with ease of transporting and unpacking.

The remaining national problem is to achieve fewer variations in data blocking structures (how individual logical records are arranged into larger blocks).
10. Formats for Archive Data

In January 1992, I attended a NAS committee meeting for the National Archives, where the concern was digital data. Will we have access to important national digital data for long time periods? Consider the formats that are needed so that present data can easily be used 50 years from now:

- The format should be as independent of particular hardware and software as is reasonably possible.

Both hardware and software will change over long periods of time. We do not want the data to become locked into structures that we can no longer read 50 years from now.

- The format should be simple enough that it can be understood and so that data can be extracted in a reasonable length of time without a lot of delay to learn the system.

- Large-volume datasets should be in formats that are reasonably compact, but still simple.

- There should be a clear description of the format and its blocking structure. The usual format should be simple enough that this description can be relatively concise. It may take more space to describe what the numbers mean.

11. The Data Railroad

To process large amounts of data, it is important to be able to move it rapidly to and from I/O devices such as tapes and disks. This means that the data flow should be as serial as possible so that the timing is not dominated by the relatively slow access times of devices. We can think of the process as being like trains moving across the country, each with about 100 cars. A lot of produce is moved in a short time, at low cost.

Consider the process of loading and unloading cars in the train. In some cases we might do this using mass production methods, one car of coal at a time. In other cases, we have more detailed work to do. In this case we move one car to a siding to do detailed work on that car. Let us suppose that the one car is like a few records in a big file of data that we want to use to prepare some graphics. The method is that we extract these records from the file and operate on them.

If the nature of the overall task is that we need to process large amounts of data, then a whole-train approach is needed. If the main task is to select small parts of large files, then we need to concentrate on methods that make it faster to access these small parts (like an airline reservation system). Such random access systems have to be reorganized periodically. This process would be much faster if they were built so that they could use the “train” approach to handle the process of reorganization.

12. What Would a Good Data Structure Look Like?

I can think of many times in the past 30 years when format questions have come up. Many people have considered what constitutes a good format versus a poor format for given types of data. Some groups have defined common formats that are supposed to solve all of the problems. About ten of these are discussed in another document.
During the past few years, I have run into more people who question the idea that one set of format constructions is best for everything. I believe that "common format" people often carry their ideas to the extreme. It all sounds good, but usually has bad side effects. The "common format" people usually agree that there should be one common format, but they do not agree with each other on what format it should be. They also seem to want to standardize all data procedures (whether or not it makes sense), but they do not want to standardize their own software.

I think that good data structures start with the idea that data is made up of a series of logical records that may be of either fixed length or variable length.

- We need a few better "packaging" standards that carry this variable length data. The packaging should keep the data under checksum protection.

- A logical record might be a satellite scan line, a grid point analysis field, or a rawinsonde report. The data could be unpacked either by using the fast "data access routines" that the Data Support Section now employs or by going through the more detailed access methods needed for interactive displays. The display routines would either decode the format or unpack elements from the existing structure. In the latter case, the data could be used by data display packages.

- Indexing methods. If reports are of equal length the indexing methods are fairly simple. Otherwise an array of indices would have to be kept if they are needed. If one usually has to access a CD-ROM to find an index, a lot of extra time will be used.

In November 1991, J.T. Young, University of Wisconsin called to ask about ideas on formats. I hope that we can all share a few more ideas without using a lot of time.

13. Toward Future Handling of Logical Records

We have noted that data are usually in some blocking structure. This automatically happens when the data are on mass storage devices. It also is a part of sending data in packets over communications networks. It is also convenient to block data for other purposes, by using structures such as IBM VBS block, Cray Block, or NCAR's Report-out structure.

For fast data processing, the data are blocked, but one logical record can be removed for more detailed handling, as noted in the railroad story above. To unpack one logical record, the program delivers the logical record to a buffer. At this point, the kernel of an NCAR access routine operates on the data to unpack it (a ship report, raob report, or satellite scan line). Under present procedure, a user takes the access program from NCAR, and uses the unpacking software kernel to decode the logical record.

What if the user has too many different access programs to cope with reasonably. Then the above process could be automated. Some groups in the U.S. are working on methods to describe formats (e.g., JPL, NGDC, University of Maryland, Glen Flierl for ocean GOFS study). Similar descriptions are used in the WMO buffer format for observations. This data description could then be automatically used to decode (or translate) the format. These methods can be used to avoid some of the pain of common formats. They would provide an automatic unpacking of formats or a translation.
However, the present method of user control has a very good attribute. The user has direct control over the naming of the variables as they come out of the unpack program. This makes life very easy for users who are doing their own programming.
Using GBYTES to Unpack Binary Data

Following is a brief summary of the steps that are needed to make it relatively easy to use data that are in a binary packed form:

- Make the subroutine GBYTES available on your computer system, if it isn't already there. This is a machine language subroutine that is available for several of the types of computers in most common use (CDC, CRAY, IBM, VAX, etc.). NCAR has listings of several of these routines; use the one for your computer. It can then be used for unpacking packed binary formats and for any other unpacking tasks. It permits one to unpack data by using FORTRAN, instead of having to write machine language for each task. Please contribute GBYTES for other computers to NCAR.

- The text "GBYTES/SBYTES FORTRAN Callable Routines for Unpacking/Packing Arbitrary Length Bit Groups" has information about the use of GBYTES and listings of the code for several computers.

- Further information about the handling of binary data is contained in "Techniques for the Processing, Storage and Exchange of Data, NCAR TN-93, Jenne and Joseph, 1974.

- Example of use of GBYTES to unpack data:
  Suppose that data in the record are packed in the following order:
  5 bits - not used; blank bits
  4 bits - month
  7 bits - year, 1978=78
  100 numbers, each 12 bits - a grid of pressure data

  Some code to read and unpack the data:

  Dimension NData (100), INBUF (50)
  C Read the record of data into the buffer INBUF
  C Unpack the month, year, and 100 values:
  Call GBYTE (INBUF, month, 5,4)
  Call GBYTE (INBUF, Iyear, 9,7)
  Call GBYTES (INBUF, NData, 16,12,0,100)

- Most grid point data has a base value subtracted and the resulting number is scaled for packing. Thus, the true values will have to be reconstructed from the packed value (in NDATA above) by applying the base and scale values.

- Use the format writeup for your specific type of data to prepare the necessary GBYTES calls for your task. NCAR has FORTRAN access codes using GBYTES for a few common datasets.

- Suppose that the data are blocked, so that there are several logical records within a physical record. Then it is easiest to first use a GBYTES call to move one logical record from the physical record to its own buffer. Then the logical record can be unpacked using GBYTES calls as above.

- NCAR is sponsored by NSF -
Section 10

Two Items Toward an Analysis of Formats
NASA Data Systems

NASA has been setting up seven distributed archive centers, DAACS, located at:
Goddard (Climate and land), about 35 people in this one.
Langley, Va (Radiation)
Boulder, CO (Snow & Ice)
SAR (Alaska)
EROS, S. Dak (Landsat, etc)
Marshall, Alabama
JPL

There are also a few associated DAACS such as the University of Wisconsin.

A data directorate for Goddard?
Rickey Rood at Goddard wrote the attached text about whether Goddard should set up a separate data directorate. Rickey told me in December 1991 that those plans have been scuttled. He argued against the idea. I think he was right.

The Goddard DAAC meeting
I am on the Goddard DAAC user working group which first met December 2-3, 1991. Rickey Rood and Al Arking (radiation scientist at Goddard) were strong advocates of the “keep it simple,” “just give us the data” philosophy. This is similar to the attached memo, Al Arking argued for ASCII formats. These formats often aren’t appropriate, but the process of ingesting data should be as simple as using ASCII formats.

The user’s working environment
Rickey Rood discusses this in the attached text. He thinks the scientist in the field will usually come to the data analysis game with his own graphics software. I feel strongly that a scientist should be able to easily use data with software of his/her choice, and not be constrained by what happens to be provided by any one system.
TO: Franco Einaudi
FROM: Richard Rood
RE: Goddard as a Data Center

(written May 1991)

Franco, you asked me to write what I thought about Goddard as a Data Center. This request was in response to your reading the viewgraphs of the presentation of the Task Group on GSFC's Role as a Science Data and Information Center. I was member of that Task Group. This memo is a substantial revision of the earlier draft I provided for the April 17, 1991 staff meeting.

I derived these views from discussions with people in the Task Group, and several individual users of the Data Center that I interviewed.

A specific question that you wanted me to address concerned the formation of a Data Directorate to deal with data related problems.

I would like to point that the Task Group did not specifically recommend the formation of a Data Directorate. Part of Jim Green's philosophy as Chairman appeared to be to provide a platform for a variety of views. The formation of a Data Directorate was one view that some members either held or felt was a valid option. Therefore it was presented. Furthermore, I understand that others at GSFC are proponents of the idea, and that their advocacy has made this a contentious issue.

I think the formation of a separate Data Directorate is fundamentally flawed. While any of several management structures could work, the formation of a separate Data Directorate appears to be the structure with the least probability of success. I base this on the premise that the Data Center must be completely integrated with and driven by the user community. The Data Center at GSFC is already too removed from the user community. Therefore, to isolate it as a Directorate is a move in the wrong direction.

The concept that the Data Center should be user driven was the most robust conclusion of the Task Group. There was one other major concern of the Task Group; namely, that people within the community of data management feel as if they are not properly recognized for their work. They feel that promotions in a Science Directorate are often based on publications, and therefore, they are not treated as equals with scientists. They feel as if scientists are reluctant to help in data management problems, because the scientists know that the data efforts will not be significantly recognized.

There is a corollary of the user driven requirement that should be explicitly stated. The Space Science community has specific data needs that differ substantially from those in Earth Science. There is a valid fear that Earth Science, and specifically Eos, will consume the Space Science data needs. I think that it is necessary that Space Science data needs be tended to by space scientists.

----------

Possible Data Center Structure.
As a given, it is the main purpose of the Data Center to facilitate the accessibility of data to the community. Therefore the Data Center first and foremost should be responsive to the user/customer needs rather than forcing the user/customer to software, hardware, and protocols that are dictated by the center. Dictated by people who are often far removed from how data is really used.

There are two approaches to data management. The first is to allow every science group to manage its own data sets (anarchy). The drawbacks to this approach are: 1) a lack of standards, 2) duplication of effort, 3) poor transport of data to other users, and 4) poor long term archiving (especially when projects end). The advantages of this anarchist approach are: 1) terrific responsiveness to scientist needs (no need to worry about bureaucrat vetoes or getting signed off on changes in a system), and 2) a system which is well tailored to the scientists home system. In essence, those people who manage the data set are primarily concerned with science, have direct day-to-day contact with the data sets, and have the authority to make major changes in the data management. Their ultimate goal is science, not data management. This anarchist system is an extreme example of a DISTRIBUTED data system.

The second approach is to manage everything from a CENTRALIZED system (Stalinism). The second approach drawbacks are: 1) top down management, 2) subsuming science requirements to hardware, software, data, personnel and system management requirements, 3) bureaucratic suppression of initiative, 4) poor communication with those individuals who directly work with the data, 5) a centralized point of failure, and 6) lack of direct interaction with the scientists. The advantages are: 1) resource efficiency (hardware, software, and personnel), 2) the knowledge and ability to implement standards, 3) good transport to other users because of the greater hardware capabilities, and 4) and good long-term archiving capabilities. As opposed to our first approach, the main goal of this centralized system is data management, not science.

Historically the bad parts of the centralized systems have proven so overwhelming that large centralized Data Centers are not well received by the community. The good parts of the distributed approach are the items that must receive highest priority. Ultimately, the measure of the Data Center must be how well the data is communicated to the users. Therefore, the data center must be willing to face up to the expense demanded by a diverse hardware environment in order assure the communication of data to the equally as diverse user community.

The Data Center should be a hybrid of CENTRALIZED and DISTRIBUTED efforts. Wherever possible, the Data Center should take advantage of ‘small’ efforts that are focused at a particular problem, such as a particular mission’s data or a collection of data focused on one discipline. Intelligent distribution of data services is very effective. It has been proven that these small data centers can work efficiently and responsively. It will be impossible to collect at a central Data Center all the expertise needed to address the science problems of a specific data set. A primary function of the Data Center should be to enable and foster the communication between the distributed data...
centers and then to the more general community.

With specific regards to the comments above about Space Science, I think that a center that addresses Space Science data needs is a healthy and necessary example of the distribution of responsibilities. I was insistent that the committee not recommend the replication of Code 930 in Code 600, because I do not think that Code 930 provides the ideal model.

It is also necessary that I comment about EosDIS. The Task Group proceeded as if EosDIS was a mandated centralized data facility, and that GSFC had to respond to this mandate. It is my understanding from the Eos Data Committee that while this was the original HQ and Project plan, that the actual implementation of EosDIS will be a set of distributed Data Centers. Clearly Eos has seen the problems of the central facility and chosen to seek other strategies. Therefore the argument that Eos requirements mandate a central facility are not correct.

The general problem of recognition is a major issue that some cite as justification for a Data Directorate. In a separate directorate there would be proper competition for promotions. I feel that this is 'equality by proclamation,' and it is not the correct way to solve the problem. It does not truly address the core issues of recognition.

One possible way to make the data work and the service work explicitly important would be to define what functions are expected of the data center people and then to recognize them appropriately. NCAR has realized the importance of group efforts and support of the scientific superstructure. NCAR explicitly defines in their Scientist I-III positions that it is understood that group work may profoundly impact personal performance.

Also management cannot effectively dictate that scientists be involved in data. Scientists who have shown an interest in data should be given rewards and incentives for their interest. Just as an example, if the Data Center provided programmer/hardware support to make the scientists data job easier, it would surely provide motivation for some. I think it is also necessary in new hires to let them know as Civil Servants there are requirements, such as data service, that are an important part of the job. The Civil Service is not an academic ivory tower.

There is also a powerful, subtle message that some feel about data service and NASA projects. Historically, projects have not given data service the same priority as space hardware and deployment. The data budget is cannibalized to meet hardware shortfalls. Basically the projects must commit to and protect the data part of their mission. Someone in the project office must be responsible for the data delivery to the community, and they must be there to protect and guide the data aspects of the mission. Ultimately, it is the data from the projects that must be used for the project to be successful. I perceive that the idea that data is important has become a more integral part of recent projects (e.g. Eos).
There are a lot of responsibilities that HQ and GSFC have committed to in the data field. Perhaps more than can be done. Given the magnitude of the problem, I think the data people have to step back and really evaluate what is their appropriate job. I think that job focuses on the acquisition of appropriate data sets, the distribution of the data sets, the provision of communication between Data Centers, and the provision of communication to users.

The job definitely does not include the maintenance and running of models. I have heard some suggest that the center should be a repository for models. I disagree strongly. I think that it leads to bad science. I think that in a world of limited resources it distracts too much from the main focus of the data center.

From the people I talked to, the data center should not be providing sophisticated graphics and visualization. Once again in a resource limited world this impacts the human and hardware factor, and uses enough overhead to make the Data Center inefficient. People tell me that the center can never provide these resources in a diverse enough fashion to meet community needs. Basically the scientist in the field is going to come to the data analysis game with their own graphics software, and will not in general choose to use a new package provided by the center. A new package that may or may not work on their local hardware. I think it is incorrect to think that the bulk of the data analysis will be done on the Data Center's computers. It will be downloaded to local systems. The graphics at the data center should be adequate to allow the center personnel to maintain quality control and check their operations.

Finally, there is a lot of talk of the Data Center providing retrieval algorithms and doing reprocessing. I think that this service by necessity must reside at distributed sites where the scientists work. Once again it is the Data Center's role to provide communication and support to these distributed centers.

The re-processing efforts to date show that we have to take advantage of the scientific heritage at the distributed sites and that direct contact between scientists and data center personal is essential. They also provide a valuable example of how data centers should be operating.

I would like to make sure that both Lola Olson and Dan Spicer are recognized for the improvements they have brought to operations in Code 930. I would like to point out that both of these people come from a science background, and they know how the data is used. Currently, however, they are simply important individuals in a system that is dominated by people who are not intimate with scientific uses of data and computers.

Finally a philosophical point. Many ideas come down from HQ. An example is Eos and EosDIS. As a practical matter what has to be done is focus and alter these ideas into a pragmatic useful format. Adopt what is good; omit what is bad. We are clearly not in a position where new ideas form in the ranks and propagate upward as a matter of course. I feel as if the Data Center at
GSFC has accepted some of the HQ ideas as unalterable. I propose that this is a basic problem in people who think about management and accountability rather than problem solving.

Feel free to distribute this memo as you see fit.

Richard Rood
05/08/91

May
Section 11

Two Items Toward an Analysis of Formats
How to Distribute Data, an Example (NODS)

The NASA Ocean Data System (NODS) at JPL was set up some years ago to help prepare some satellite ocean data, and to help users access it. NODS was to serve the Ocean Research Group at JPL and people at universities. They developed a rather elaborate on-line access system, but did not have many datasets. It was funded out of NASA Info Services Office (ISO), at a rate that peaked at about $2 million per year. They were instructed not to develop "new" datasets - SSMI Sea Ice was an anomaly -- but to ingest datasets produced by others and provide catalog and subsetting services. The idea was that the NODS data system would be developed by ISO and then moved to research budgets. This was done, but there was frustration over this large a chunk of money taken from research funds. Most of the cost was associated with the creation, maintenance and operation of software for on-demand subsetting and browsing of data.

The front page of the Cryospheric Newsletter (attached) gives the present status of thinking. My impression is that the approach may be going from one extreme of preparing detailed data subsets automatically to the opposite extreme (of not preparing subsets). For some datasets it makes sense to prepare subsets in a few simple, standard ways. This is usually easy and inexpensive to do. One should not get too elegant or offer too many options, or the cost goes through the roof.

The NODS system has about 150,000 lines of code that is on a VAX. It would be hard to migrate the code to another computer.

This includes review comments by V. Zlotnicki and E. Enjoku, NASA.

Note: NODS policy (no data is sent electronically). Info Sep 1989
**CDMS Notes**

Cryospheric Data Management System

Issue no. 3  
January 1989

NASA OCEANIC PROCESSES PROGRAM REVISES DATA DISTRIBUTION PHILOSOPHY

In 1988, the NASA Oceanic Processes Program (OPP) following the recommendation of the SODSSWG*-NODS Advisory Panel (SNAP) decided to revise the method in which data will be distributed to their scientific investigators. Originally, the NASA Ocean Data System (NODS) was funded to maintain a centralized archive of oceanographic data. The NSIDC NODS duplicate system, for example, will be responsible for maintaining the archive of SSM/I data products for the polar user community.

In the original model, access to data holdings residing at these archive centers required the investigator to access an on-line menu-driven interface. The archive user contact would be via an electronic network (such as SPAN or direct-dial) to extract, display, or order the data.

*SODSSWG = Satellite Ocean Data System Science Working Group.

---

**TABLE OF CONTENTS**

This newsletter is published to provide useful information to the research community about current events at NSIDC. The focus of CDMS Notes will be the CDMS, current ice and snow data products available at NSIDC, and the status of applications using these products.

| NASA Polar Oceans Branch Revise Data Distribution Philosophy | 1 |
| NSIDC to Distribute SSM/I Grids on CD-ROMs | 1 |
| AVHRR HRPT Data Available from AFC | 3 |
| Geolocation Correction to be Applied to SSM/I Gridding Procedure | 4 |
| Wantz Data Tapes will Replace SSSD Data Tapes as INGEST to CDMS SSM/I Archive | 4 |

---

A review of on-line access and distribution methodologies was made by the SNAP and NASA OPP management. Their assessment is as follows:

- The distribution of customized data sets from an archive is an expensive use of computer and human resources.

- On-line access to a center's archive over current communications technology is tedious.

- On-line access to a central archive requires a high degree of training which demands the time commitment from the user and staff resources at the data center.

The following summarize the resolutions to their findings:

- Distribute all of the data to the investigator and provide the tools necessary to extract, subset, and display the dataset on inexpensive workstations at the investigator's home institution.

- The data center is responsible for the management of the archive including preparation of data for bulk distribution to the community.

- Distribution will be made on low-cost media; the investigator will be able to access the data via an inexpensive workstation (286, '386 AT, Macintosh-II, VAXstation, SUN, etc.)

**NSIDC TO DISTRIBUTE SSM/I GRIDS ON CD-ROMs**

The SSM/I Compact Disc Read Only Memory (CD-ROM) will be mastered using the ISO 9660 industry standard. These disks will be compatible with the MS-DOS operating system. To support compatibility with the VMS operating system, extended attribute records (XABs) will be supplied onto the disks. The XABs will be transparent in the MS-DOS environment. Hardware support for Macintosh computers should be available by mid-1989. Apple is currently testing their version of the extensions for MS-DOS software. This software will provide access to the data residing on CD-ROMs for the Macintosh computers. Application software for VMS and Macintosh operating systems will have to be developed to support the release of the SSM/I CD-ROM.

---

National Snow and Ice Data Center, CIRES, Campus Box 449, University of Colorado, Boulder, CO, 80309  
Telephone: (303) 492-5171; Telex: 257673 WDCA UR; Telex number is accessible via Telemail
Section 12

Two Items Toward an Analysis of Formats
Dr. Shelby Tilford  
Director, Earth Science and Applications Division  
Code SE, OSSA/NASA HQ  
600 Indep Avenue SW, Room 219  
Washington, D.C. 20546

12 February 1992

Dear Dr. Tilford:

I am writing this letter because it looks to me as if NASA may make some data format and other data decisions (too quickly) that could have consequences such as:

- Take high volume data in formats prepared by instrument teams (I think these have usually been good), and change these to fit the particular ideas of certain “data managers.” In about 1990, I talked with Greg Hunolt about this, and he did not think that it made any sense either.

- Take high volume data, choose formats that blow up the volume, increase the costs to store data, and make it more difficult to exchange data.

- Require scientists to install large data delivery software packages in order to use any data.

- Require PIs to learn large systems, whether or not there are easier methods.

- Choose formats that do not permit any flexibility, do not accommodate WMO standards like Grib and Bufr, do not have the capability to reasonably store gridded data, do not handle variable length structures, and do not permit engineering ingenuity on compression.

- Provide system-defined data use, not the flexible use that is needed.

- Spend 95 percent of data management money on this sort of data access, when at least 90 percent of the scientists would find simpler, lower cost methods more helpful.

One example of good data work is that JPL is achieving 3 to 1 compression on large amounts of planetary data. This makes it possible to create hundreds of CD-ROMs at affordable cost, and get it to the scientists who need it. We should not create any policies that increase such costs.

There are some scientists who are primarily interested in the system aspects of computing and data, but I believe that up to 95 percent of working PIs would agree with the following:
We want easy access to the data with low learning times. The low learning time needs emphasis. It should be quite easy to ingest data for use in a program, even if a PI only uses the procedure once each six months.

In the data "shop" we want to be able to talk with someone who knows something about the mechanical and scientific attributes of the data.

We want to use the data for calculations and in display systems that PIs choose, not that are dictated by the data system. Some system browse is reasonable, but it does not give more than 10 percent of what PIs need.

A key requirement of good computing working environments for scientists is that it must be easy to obtain data, and it must be easy to insert data into various software tools. Most of the latter tools will be from other scientific groups and from commercial sources.

The PI will choose software on the basis of usefulness, ease of use, and cost.

Scientists know that there are formats that are trivial to learn in order to ingest the data they need. They fear that they will be forced to use something much more complicated and cumbersome in order to obtain any data.

There is a tendency for people to design systems that are costly, inflexible, and hard to use, with the explanation that the users requested it.

The technology that is good makes our life easier, is not obtrusive, and helps us to do what we want to do. Many methods will claim great attributes, but they still don't deliver the goods. The draft Interagency Plan calls for an evolving data system. This implies that we don't freeze any more decisions than we have to, and that we can imagine a path that leads to future improvements.

I think that the present problem is that the emphasis is being put on common formats, rather than on a few standard data envelope structures that make it possible to design systems that are much more flexible. When we review the history of claims to common formats, we note that new ideas (which are sometimes improvements) move through the field about each three years. And this is only for one or two disciplines. I talked with Dixon Butler for a short time at the Data Forum in January. He noted that it probably costs at least $160K-$180K to support a common format, so you do not want too many. This is understandable.

One problem that we have had in the last 10 to 15 years is that the issues of data use get lumped together too strongly with the issues of data display. This probably happened for two reasons. One is that displays are needed and the other is that the people who created several data systems seem to have been most interested in data displays. The needs of both data use and display have to be considered, but the displays should not drive the whole system. There are display systems (one example is at the University of Maryland), that are very good, yet they still have graceful ways of importing and exporting data. It is interesting that all of the several best display systems (Mc Idas, Gempak, University of Maryland, CRD tools, etc.) have their own internal formats which are not any of the formats being discussed.

I think that one of the problems of large organizations concerning data is that the main advisors to high-level managers have interests in computing and software systems, but too often their agendas and experience can be quite different from that necessary to produce the priorities and working environments that most PIs need to accomplish their research.
Dr. Shelby Tilford  
12 February 1992  
Page 3

I believe that one of our national problems is that “data” has been hyped too much. I am glad to see some real interest in data, but I think that the present situation can easily lead to many budget games rather than a more narrow targeting of resources toward the most pressing problems. It becomes easy to build white elephants. Too much hype makes it easy for people to put pressure on managers like you, and I suspect that it becomes more difficult to sort out priorities.

There are interesting developments around the U.S. that could be used to avoid many of the common format hassles. The idea is to take a logical report such as a satellite scan line, a model gridpoint field, or an upper air rawinsonde report. Then put these reports into an envelop that permits fast access for calculations, or delivery to display routines. Do not convert everything to one standard format up front. Do that only when and if it makes sense. At a NAS National Archives meeting in late January 1992, I had a chance to talk to Ralph Bernstein of IBM about this. It is clear that this is his concept, without prompting from me. He chaired the NAS data panels a decade ago (CODMAC reports). Some of these procedures have been used for some time; some are now in development by a few groups.

Joe King recently told me that he has NASA headquarters responsibility for format issues. I saw him again at a National Archives meeting about January 30. I will send him a copy of this letter and some technical material about formats that may be useful to help sort out the problems. I think it would help to sit on any format decisions until the issues are much better sorted out than they appear to be at the present time. Perhaps it would be useful to have a meeting where the format groups, format analysis projects, data compression people, and data format translation people all listened to each other for two days.

As NASA gets closer to defining the new data systems, it seems to me that it is very important to avoid falling into the NIH situation (some very costly computing decisions) that was recently reported in the 16 January 1992 Nature Magazine (article attached). The data issues are complicated, and I wish you well in sorting them out.

I think that the NASA/NOAA Pathfinder dataset program is very good for science. I also think that the PI teams asked to sort out data rate problems during 1990 and 1991 came up with reasonable results. I believe that there are technical opportunities to handle high rate data and still reduce costs that need to be considered.

Bob Schiffer asked me to be a member of the Goddard DAAC panel. At the first meeting in early December, I was pleased to see that Al Arking and Ricky Rood (from Goddard) were also there. At some point, it could be useful for you to have a talk with them.

Sorry to bend your ear this much. I am looking forward to a lot of interesting NASA space missions in the coming decade, and a lot of good research.

Sincerely,

Roy Jenne  
Senior Scientist

RJ:ob  
enclosure  
cc: Joe King  
Ferris Webster
Cover Memo

From: (O:GSFCMAIL, SN:KING, FN:JOSEPH, I:H, SITE:GSFC)
To: R.JENNE/omnet
Subj: Formats, etc.

Roy, seeing Shelby Tilford today reminded me of your formats letter to him of a couple of months ago. You cc'd me on that also. Did you get a reply from him?

I recently created the file below. I would very much appreciate any comments you may have on it.

(This is my first attempt at going from GSFCMAIL to OMNET; pls acknowledge receipt.)

Best wishes,

Joe King

Forwarded message:

Posted: Fri, Mar 27, 1992 7:21 AM EST       Msg: GJJC-1706-3626
From: JKING
To: dsawyer (c:usa, pub:telemail, pvt:nasamail, o:nasa, un:jlayland)
CC: jking
Subj: stds program, etc.

Don, as you may already have heard from Jim Layland, there was much interest in the our standards program, and in the upcoming "formats workshop." Below is the product of an action item I picked up. I welcome input from both of you on this, before I forward it on to others early next week. (That's "... picked up at the ISB program review just ended.

Joe
NASA Format Study

ISB STANDARDS PROGRAM & PROCESS FOR "STANDARD FORMAT"
RECOMMENDATIONS

by Joe King, NASA
March 27, 1992

ABSTRACT: This paper gives: a high level statement of the objectives and activities of the Information Systems Branch (ISB) standards program; an outline of a now-starting year-long process to develop recommendations on "standard formats" for the OSSA scientific and management communities; a brief closing statement on the directions of the ISB standards program.

Introduction & Program Overview

Several "information system technologies" have the potential of greatly facilitating the OSSA science endeavor. Examples of such technologies, specifically relevant to OSSA's data management program, are data formats and data media. Each technology typically has several "implementations." Examples of implementations of data format technology (or, in more normal language, examples of data formats) are SFDU, FITS, and CDF. (We use the term "format" liberally throughout this paper.)

When an implementation of a technology becomes sufficiently widely used, it is referred to as a de facto standard. When the full specification for a given implementation is passed through a formal accreditation process of a standards body (e.g., ISO, ANSI), that implementation is referred to as a de jure standard (e.g., an ANSI standard).

Some entities (persons, groups) support the development, knowledge of, and use of standards. Other entities, by virtue of their organizational authority, mandate the use of specific standards in their programs; such standards thus become "mandatory standards" for the affected programs.

The objectives of ISB's standards program are to:

1. Support knowledge of information system technology implementations on the part of OSSA scientists and managers;
2. Advise OSSA discipline and/or project managers in their selection of technology implementations to become mandatory standards for their programs;
3. Make recommendations to the OSSA science community on the use of specific voluntary standards;
4. Support the use of specific standards, as by conformance testing for format standards;
5. Support the development of specific standards;
6. Support the formal accreditation of "OSSA standards."
There are two formally constituted (i.e., funded) elements of the ISB standards program. These are the NASA/OSSA Office of Standards & Technologies (NOST) at GSFC/NSSDC, and a standards group at JPL. The primary focus of the JPL group is the development of the Standard Formatted Data Unit (SFDU) in the context of the international CCSDS program. NOST shares in this activity, and in addition covers a wider range of activities.

One key NOST activity is the management of an online information base, the Standards & Technology Information System (STIS). While NOST staff effect some NOST population, it is a goal of ISB and of NOST that information should flow into STIS from experts throughout the OSSA information system environment.

**Formats Assessment/Recommendation Process**

It is generally recognized that wise use of the right data formats will significantly benefit the OSSA science endeavor by enhancing data transportability, generic software development and use, etc. However, in today's OSSA data management environment, there exists confusion relative to apparently competing data formats. In fact, some format standards involve rules for organizing observational (or equivalent) data, while others consist of rules for organizing metadata only.

In order to help science community members select the best format(s) for their applications and to help OSSA discipline and project managers assess the merit of mandating specific standards, NOST is beginning a multi-stage process to clarify the "standard format environment." Objectives of this process will be to:

1. Characterize individual formats in terms of their objectives, histories, uses (e.g., used by what projects), supporting infrastructures (people, especially), associated software to exploit the format, evolution directions (and how determined), etc. (June 1992)

2. Summarize the relations, complementarities, etc. among these formats.

3. Summarize the relative strengths and weaknesses of the formats when considered for various applications.

4. Identify and summarize efforts to build bridges (or transformations) between formats.

5. Identify work needing to be done to evolve individual formats, or to build bridges/transformations, to meet user requirements.

6. Develop recommendations to the science community for the voluntary use of particular formats for particular uses.

7. Develop recommendations to management relative to the mandating of particular formats for particular uses.

It is expected that this process will require at least one full year, and that this process, or some abbreviated version thereof, will have to be repeated periodically as both requirements and technologies evolve.
The process will begin with the compilation of a report addressing points 1 and 2 above, for the following formats: HDF, CDF, netCDF, FITS, SFDU, PDS labels, (get others). This report will be the product of efforts of one or two experts per format who will come together in June 1992 at Goddard.

This report will be disseminated to various scientific, technical, and management persons, who will be invited to a workshop to be held at Goddard in the fall of 1992 to address the remaining points above. Candidates for participation in this workshop will be identified through the OSSA Discipline Divisions and their Discipline Data System personnel. Additional persons identified by ISB as having the potential and willingness to contribute significantly will also be invited. Format users will certainly be valued as participants (fall 1992).

It is intended that this process be coordinated with like discipline-specific or project-specific efforts, as for instance with EOSDIS formats assessments for EOSDIS purposes.

**Program Direction Determination**

GSFC/NOST and JPL are currently developing a plan for the evolution of the OSSA/ISB standards program. This plan will add flesh to the skeletal vision offered in the Introduction of this paper. It will identify what is to be done, why, and at what rate and resource level, and it will offer some sense for the priorities among the program elements. It may involve the undertaking of processes for other technologies similar to that just outlined for standard formats. The plan will be shared with the OSSA Information Systems Management Board for its comment and endorsement. In this, the ISMB represents the intended benefitting community of OSSA researchers. The plan will also be shared with Discipline Data Systems scientific and technical personnel for their comments and insights.
Section 13

Data Management for Climate and Global Change
(Feb. 1990)
Data Management for Climate and Global Change

We have a small data section (six people) at the National Center for Atmospheric Research (NCAR), whose job is to ease the problems of data access for research in Meteorology, Physical Oceanography, and overall climate studies and modeling. The group has one of the largest archives of data (over 350 datasets, and a volume over 16 trillion bits). Users with access to the supercomputers at NCAR (400 users at NCAR, 650 at universities) can also readily access the data online. Simple data access programs are provided to unpack the archived data, and present arrays ready for calculations. Learning time is very low.

Other users (from around the world) obtain data on magnetic tape. Most of our data access software is transportable to computers of various types. We now have an activity to provide more transportable access software along with the data (especially for the packed binary data).

1. Design for Evolution of the Data Systems

It is essential that data systems be conceived so that they can gradually accommodate good new ideas and ignore poor ones. This means that good systems are designed to cope with diversity, even though the amount of diversity is kept limited. The access to data and information about data should not be dependent on particular hardware or software systems.

DISCUSSION: When we consider the history of data systems, it is noted that there have been many changes on time scales of two to five years. There have been changes in data storage systems, concepts to define formats, and in computing capability. It is unlikely that a data system defined today will remain constant for the next decade. The design problem is to develop data systems that are flexible enough that the data and also information about the data (metadata) can be readily saved for the next 100 years even though changes occur. Thus, the hardware to store data will change and the software to help access data will also change. The basic data archive should enjoy relative stability so that errors are not introduced by frequent conversion and handling.

It is bad practice to change all formats in an archive every time a new idea comes along. Also, the metadata (information about data) must be structured so that it can be readily used by future systems as well as present ones. Thus, a “data system” should be conceived of as an entity that operates on data and provides easy access, not as an entity that owns the data.
2. Problems of Data System Design

It is far too common that data systems are only designed to accomplish a narrow range of functions that the designer thought of, or had time to accomplish. Then they are sold as being good for all functions of meteorology, astronomy, curing rheumatism, and climbing mountains. They are often conceived in the model of what is needed for an airline reservation system. They often are not flexible enough to handle input formats received from elsewhere in the world. This is not the model of what is most needed for scientific computing. For this, there are several needs:

- Prepare the necessary data in reasonable structures (not necessarily "common" structures). For high volume data, emphasize tightly-packed, efficient structures that control costs. Emphasize technical developments to decrease the actual cost of data transfer and access.
- Provide simple data access routines that handle the task of obtaining data from storage, and unpacking it so that it is ready for calculations. These must be efficient, and have a short learning time (from 5 minutes to 2 hours).
- Optimize data input methods so that the user can easily do his/her own computing using a higher level language (most commonly Fortran).
- Provide tools for graphical displays so that the user can easily prepare displays the way he wants them. This includes tables, plots, and VCR motion film.
- Encourage the preparation of a few "canned displays" that are commonly used in part of a discipline area.
- The world will have a number of data manipulation tools that are focused on certain tasks. Examples are: Lotus 1-2-3; a DBMS; and Unidata (meteorology). It must be easy to take an array of data from Fortran calculations and move it to one of these data manipulation software packages.

3. Relevance of Data Management Activities

It is sad that too often meetings about data management are not relevant for the main data needs of the users. In the Data Support Section at NCAR, we listen to the needs and troubles of users, day after day. Some examples are: (1) For mesoscale models, we need higher resolution datasets of earth vegetation cover, lake extent, and elevation. Can you help us? (2) We want to use modern analysis methods to reanalyze all of the world's surface land, ocean, and upper air meteorological data from 1958-on. Can you help us obtain the necessary data? (Yes, but for minimum standards of data completeness, this will be a very large task. (3) Can you help us obtain a lot more data over Africa? (4) You are just emphasizing access to data at NCAR, or delivery of data on half-inch tape (and some by communications). We want to have easy access to large amounts of your data at our university department, at low cost, using technology such as CD-ROM disks, DAT tapes, or EXAbyte. We want data access software that makes it easy to make calculations using Fortran.

In many cases the focus of data management discussions is on data browse and display, not on the central problems that we and users face.

4. Complaints About Data Access

Some complaints about data access and data systems can be paraphrased as follows. The last CD-ROM we got has about 200,000 grid analyses showing meteorological fields over the Northern Hemisphere for the past 30 years. The access software was set up so that we had to interactively access each field. We wanted to do serious computing using thousands of fields. We did not want to waste months of our personal time to accomplish what was most appropriately done in an easy batch mode (This is now fixed). We want to make calculations without fighting a system, and without paying a huge cost of system
overhead. Then, we want to prepare displays of the results the way we want them to look, not the way a system decides they should look. We welcome tools to prepare these displays, but don’t tell us that we should spend two weeks to learn your system and memorize hundreds of cryptic commands before we can do useful work.

5. Catalog Systems for Information About Data Availability

At the national and local levels users must have easy access to information about major datasets and where they are held, but only the most necessary information is needed at the national level. More detailed information has to be readily available at the data centers. A national data catalog system, coordinated by NASA, is under development. Nearly 1,000 datasets are already described in the system.

A catalog system will often give the user more details than can be assimilated. For a given subdiscipline area, we also need brief overview documents (online and text) that present the most important information. Examples are: satellite sounder data, data on US energy statistics, etc.

The preparation of these overview documents will also encourage people to think of what data should be prepared into datasets.

6. Data Structure

Over the years people have found it very useful to organize a given dataset using one to several files, each composed of physical and logical records. To efficiently handle many types of data, we must retain the ability to easily store and retrieve variable length records. Given this need, why do people periodically talk as if their new “systems” will no longer support these structures?

Data structures that we use for data processing make it possible to reorganize data at very high speeds, and to quickly select data subsets. We are not willing to give up this capability. Example: we have a dataset with weather reports from the world’s ships 1854 - 1980. It has 72 million reports (each with about 40 numbers). We can completely invert this file (volume five gigabytes) in four hours of CRAY-1A time. Most people now agree that to handle large datasets, it is important to keep the data in a simple file structure, and to keep the data flow fast.

7. Problems of Bulk Data Packaging

One of our key services at NCAR is to be aware of the problems of data availability (or access) that people are having, and to help solve these problems. In many cases, problems arise because low volume data and high volume data are together in one file. Some examples are: (1) Analyses and observations (now on 50 tapes) were hidden in 2,000 tapes of data; (2) One set has monthly averages integrated with 22 tapes of daily world station data, such that the small volume of monthly data is hard to access; (3) A satellite dataset has 800 tapes, yet the main data that people need could be put onto about two tapes. We are always solving a few problems like these, and sometimes encourage other groups to help.

In addition, for the larger datasets, we maintain very fast software to select data subsets based on time, region of world, variable needed, etc.
8. System Cost and System Needs

We need to strive for cost effective data systems. An analogy is useful. In a family, it is usually easy to justify the purchase of an automobile. We need it for work, shopping, pleasure, etc. How many of us, therefore, purchase a Mercedes? Building data systems often present us with similar choices. We need storage, communication, media for data transfer, access software, display software, etc. We need to choose these resources to maximize the main goals:

- Prepare the data necessary for research.
- Operate observing systems to obtain necessary observations.
- Provide easy access to required data for research (at both local and remote sites).
- Decrease the time needed to access data, and to learn systems. We want results here, not slogans.

9. Data Activities That are Needed

The following national data activities are needed:

- Prepare (and gather) the needed datasets and selected subsets. This is the main problem retarding the sciences.
- Promote the use of low-cost technology (eg CD-ROM and DAT) to distribute data. Provide portable data access software with the data. Promote other technology to reduce the real cost of giving access to high volume datasets.
- Encourage the development of lower cost mass storage systems that have one-time cost categories of (1) $700 (use CD-ROMs or DAT-manual mount), (2) $3,000, system with small carousel, (3) $20K, (4) $125K.
- Develop better solutions to the “data packaging problem.” Let users define their own formats. But develop a few variable length blocking structures to “package” the data. These would be efficient, have checksum protection, and could be used for transfer of data between computers as well as for local storage.
- Prepare translators to change data from “simple array structure” to a structure needed by one of the interactive data manipulation packages (an image analysis package, a PC publisher, Lotus 1-2-3, etc.).
- Many scientific groups prepare a battery of software to help work with their type of data and display it. Promote activities to gather selections of this software.

10. Papers of Possible Interest

- Data Management Methods; Data for Europe
  This paper discusses data strategy, data progress, and gives a history of selected data ideas over the last 20 years.
- What Types of Data Access do Users Need?
  A few short papers review how scientists use data for both calculations and browse functions. This includes examples, a paper on “Levels of Data Support,” etc.
- Planning Guidance for the World Climate Data System, 1982
  (by Roy Jenne)
  Surveys data requirements. Lists the variables. Discusses data management strategy. It is document WCP-19, available from WMO, Geneva. The planning for climate includes most of the variables needed for “Global Change.”
- Other Papers
  A list of documents that discuss selected data types, available inventories, data management, etc. is available.
Section 14

Some Attributes of Desirable Data Structures
(May 1989)
Some Attributes of Desirable Data Structures

A. Data structures that permit mixing of reports

1. Variable length logical records are often desirable where some reports require 40 bytes and others need 200, for example.

2. Two different types of reports (apples and oranges) usually will have different lengths. It is nice to be able to mix apples and oranges together when desired, without any problems caused by the differing lengths.

3. Need to be able to easily sort apples, oranges, or a combination of apples and oranges.

4. From a data blocking standpoint, it should be easy to select a subset of data. If a given report is not selected, it should be quick to skip over the data. Saving a given report should be a matter of saving its string of bytes. No information about the contents should be needed.

   Some schemes require handling of all data even if it is not of interest.

5. There should be good error control on the data (a checksum on the blocks of data).

6. On a read of a logical record, a user should be able to obtain the report length and error status.

B. Do present systems satisfy the above? No.

1. In present systems, the use of fixed blocking of constant length records is often forced. When variable schemes are available, each system often uses its own unique method of implementation.

2. Error control may be limited and error recovery is even more limited.

C. What is needed at the storage level?

1. The storage system should need to know very little about the data it keeps. It needs to know a name and an owner. There should be passwords to prevent unwanted writing.

2. The data should be just an unknown string of bits. The system knows where the data is and the amount of data.

3. The string of bits is organized into logical records, physical records and files.

4. The data will be further organized into blocks by the system.

   — A block might have length 32 KB, 16 KB, 4 KB, etc.

5. Tight error control is needed. This means a checksum on each block of data.

   The alternative is to do much of the data blocking outside of the system and to make sure that there is a checksum on each "physical record."
6. Some aspect of a storage system will periodically need to be changed. The system design should permit gradual evolution without having to change the actual data in the datasets. We do not rewrite all the books when we move a library.

7. If new media technology is obtained for the archive, it should be possible to put the data onto the new media without bothering users. They should still see the same string of bytes as before.

D. Data routing level

1. Need to know the destination of the data
2. Do not need to know the format
3. Need error control

E. Long-term storage of data

1. Media will change. Data management systems will change. One rule is: Never require changes in all datasets for a new data management procedure to be effective.

2. If people have to know about the data formats of each dataset in order to put the data on a new media, much time would be needed, mistakes would be made, and data would be lost. Avoid this case.

F. Data definition information

We have argued that at the data handling levels of data storage and data transmission, the logic that is used should not have to know anything about the internal format of the data. But when data are used, the format needs to be known. There are three main ways of handling this:

1. Put a format code number in each report. Use this code number to chose the proper program routine that knows how to unpack the report. This is a fast and "clean" approach.

2. Put a format code in each report. Use this code number to find a data description packet of info. This packet tells a general program how to unpack the report. Such data description packets can be included on a data tape or a data transmission. The packets only have to be given once, regardless of data order. This method works very well when the number of formats is very small compared to the number of reports, but is more restricting and harder to implement than #1.

3. Put a data description packet in front of each report. If there are ten reports with the same format, the description packet is needed only once.

This concept is being used in the ECMWF-WMO format procedure called Bufr. If there is only one report of a given type, the volume increase is a factor of about 2.7. Also, it appears to me that this structure would be hard to directly sort, because the program would have to keep track of description packets and would have to insert and subtract packets as necessary.
For various reasons, I think that the approach given in items 1 and 2 above is usually simpler and more efficient.

G. Data access from optical drives

There are two methods for data access. One is to find the desired data file on the CDROM or other optical disk, and rapidly transfer the file to the computer disk drive for further processing. This is then similar to loading a file from a tape, but it is faster to get to the start of a file on the disk (compared to tape). This method will usually give faster data flows if a significant fraction (such as 10%) of the data will be used. A variation of this method is to read the data serially as above, but not stage it to the computer disk. Just use it directly from the external optical disk.

The other method is to directly access each report (this means that one or two disk sectors of data are read, and the report is extracted). This is the fastest way to get at small amounts of data, but it usually isn't good practice for significant data processing.

H. Need for hardware standards for data exchange

We will always need three or four world media standards (tape or disk) that allow us to send data around the world. The half-inch tape standard has been very useful. Record structure (not detailed format) standards are needed to permit convenient data shipment via communication links.

It should not cost more than a few thousand dollars to buy a low-end drive to read standard media. People are now getting used to paying $700 - 1,000 for a CDROM reader. The cost of EXAbyte tape readers (installed) has been about $7,000, but one recent bid (April 89) was for $4,200. It is possible to buy a low-end drive to read half-inch tapes for $3,000-5,000. The biggest market will be for equipment in this price range. These hardware costs will define the media most commonly used for data exchange between many users.

Some optical drives will come in at prices of $3,000 - 6,000, and will become important for data exchange if a few standards are achieved.

PC floppy disks are now an important form of data exchange for small datasets. A number of datasets (or subsets) are rather small.

The IBM 3480 technology could be an important medium for data exchange, but at present costs for tape drives, it will be limited to a few high-end users.

I. Format structures

Data for long-term storage should not be kept in some system or DBMS structure that is likely to be out of date in five or ten years. It should be in some format that is clearly described. If a given system format is needed, this can be created when the data are read in.

J. Future

Various data centers and research groups will gather data, and put it into organized datasets.
Some of the data will be obtained by dialing in to the central system:

- The most important function needed to achieve this will be an easy, cheap way to deliver a small- to middle-sized file from the central computer to the user's computer system.

- A data center should have inexpensive ways to extract a few commonly used types of data subsets.

- Simple data decode software should usually be delivered along with the data.

The user will get much of his data on a few types of media: tapes, CD-ROM, EXA byte, small optical drives, etc.

K. How will data be used?

Will people dial-up a central data system to browse inventories or to look at a small selected set of browse data? Some of this use will happen. We should not skew priorities heavily in this direction. A certain amount of effort to provide this capability is worthwhile, if it is simple, and if the systems have a low development cost.

The move now is to distribute more data once it has been gathered. Tapes, disks, and CDROMs will be used.

Different people will offer software that will operate on given datasets. A "canned" system has a limited amount of flexibility. For many types of research, people will want to make their own calculations and create their own plots.

L. What is needed for catalogs and inventories

1. Contact points

People will often know about at least one point of contact for data within their area of interest (such as meteorology, oceanography). But as research becomes more interdisciplinary, they need help in finding the right contact.

- Therefore our printed and on-line systems need to help people find contact points for broad discipline areas, and for more specific concerns.

- Our on-line systems need to show some of the best published summaries about available data.

2. Datasets

The Federal Systems need fairly simple info about significant datasets. This is coming along.

3. Lists

Consider including various lists that give alternative ways for finding out about datasets.

Also point to summaries that give an overview of "all satellite x-ray data," "the world's tide data," etc.
4. Inventories

Most people will work in a discipline area. They may already know where the big datasets are. In meteorology, a big dataset may have data for 3,000 or more stations for about 100 years. The user will want info about when specific stations started and when there are gaps in the record.

At NCAR, we make a number of generalized inventories that meet much of this need. A list of these is available. They can be browsed on-line. At some point, we will also put the inventories on a CDROM.

Some aspects of data coverage are best shown graphically. We need a good way to put a compressed version of a scanned image on-line.

5. Answers to common questions

A person writing a book for fisheries recently asked where he could find published plots of world sea water temperatures, and world air temperatures. Some of these questions are common enough that we may put some info on-line at NCAR. In general, there needs to be some people in each data center with a scientific background, so that they can help answer such questions.

6. Empty shells

There are many catalog/inventory systems around the world that were described to me (in 1980) as "empty shells." They have fancy names, perhaps lots of advertising, but little content. The moral to this story is that the content is more important than the system, though both are necessary. The system needs to be able to include new types of content if that proves desirable.

7. Other text

Additional information is available about catalogs and inventories.
Data Structures

Many Systems Now
- Fixed record size is often forced
- It is often inefficient
- Error control is usually limited

What is Needed
- Logical records
- Variable length, blocked together
- Can easily sort, if desired
- Can mix apples and oranges
- Easy to select a subset
- Checksum on blocks

Storage Level
- Don’t need to know format
- Only need to know owner, and amount of data
- Datasets and data info are just strings of bits
- Organized by logical records, records, files
- Need error control
- Must be easy to move data to new media

Data Routing Level
- Don’t need to know format
- Only know destination
- Need error control
Section 15

Data Formats for Historical Weather Data on CD-ROMs (Dec. 1991)
Data Formats for Historical Weather Data on CD-ROMs

Dr. Harold J. Edmon
Department of Atmospheric Sciences
University of Washington
Seattle, WA 98195

20 December 1991

1. Introduction

The purpose of this paper is to lay out the choices of data formats for historical weather data on CD-ROMs. I will start with a general review of the different options as well as the different data types. At the end I will come up with some conclusions and recommendations.

2. Choices

The choices at the top level boil down to two: leave the data in its raw or transmitted form, or convert it to an accepted common data format before placing it on the disk. The advantages to the raw format are:

1. No need to convert the data before mastering.
2. Data is usually in a very compact form.
3. Can use whatever decoders are available to read the data (i.e. Unidata LDM).
4. If a decoder error is found, the data on the disk is unaffected.

Disadvantages to this approach are:

1. Different decoder or access software needed for each data type.
2. Hard to slice the data differently than it was sent. For example: to pull out a time series for a single point in grid point data stored in GRIB requires partially decoding every grid in the series.
3. Supplementary data may be required. For example: station locations for surface and upper air observations that include any movement of stations.
4. Documents are necessary to describe each data format.

Advantages to a common data format are:

1. All data is in the same format.
2. All data can use the same data access routines.
3. Can add in meta data (data about the data) such as units, station location, etc. Thus the data can describe itself, reducing the need for supplementary documentation.

**Jenne note:** I do not think that meta data is a good argument. Whether NetCDF is used or not, the meta data must be available. Also the meta data expands over time.
4. Can be easy to take arbitrary slices of the data.

Disadvantages to this approach are:

1. The data must be converted to this format.
2. Data volume can expand.
3. Raw data must be provided for data sets whose decoding is difficult and subject to error or interpretation.
4. The common data format chosen may be incompatible with previously developed software that uses a different format.

Some of the advantages/disadvantages are different for various types of data. Grid point data is usually not subject to decoder errors whereas hourly aviation data decoding is prone to errors in the original data as well as errors in the decoding that sometimes only a human can figure out. Thus I believe it is important to consider the data types separately when considering which format to use.

Another factor to consider when looking at data formats is possible expansion or contraction of the data volume. This consideration is important for two reasons. First is the amount of data you can put on a single CD-ROM. The more compact the data, the more you can fit onto a disk, thus the less a user needs to swap disks. The second consideration is the speed of CD-ROMS. Currently, CD-ROM drives are slow to seek (upwards of 1 second) and slow to read (approximately 150,000 bytes per second). Thus the more data you have to read, the slower you will get it.

3. Common Data Formats

Before talking about the different data types, it is important to talk about several of the common data formats that are currently popular. The two that come up most often in discussions of data formats are netCDF and E-BUFR. There is another format that is widely popular, HDF from NCSA, but HDF is very similar to netCDF, so that much of the discussion about netCDF is also applicable to HDF.

3.1 NetCDF

"The Network Common Data form or netCDF, is an interface to a library of data access programs for storing and retrieving scientific data. The netCDF is an abstraction that supports a view of data as a collection of self-describing, network-transparent objects that can be accessed through a simple interface." (NetCDF User’s Guide, version 2.0, UPC). The library includes functions that can take arbitrary slices of n-dimensional arrays, called hyper-cube access. The netCDF library is available on a large number of computing platforms, making it attractive as a data format for CD-ROMs. NetCDF is network transparent since it is implemented on top of a layer of software for external data representation (XDR), which is also used by a widely available set of software for sharing files across networks (NFS- Sun’s network file system). Thus one advantage of the software is that most of the machine dependent work is already done, since virtually all computer manufacturers support NFS. In addition, there is a growing set of software that can use netCDF files directly:
1. NetCDF operators under development at UPC.
2. MATLAB through an interface developed at USGS in Woods Hole.
3. A model output plotting program (MOPS) under development at USGS in Reston.
4. The NCSA software (e.g. XDataSlice) is being modified to also read netCDF files.
5. PolyPaint from NCAR.
6. AVS via a module written by USGS.
7. Spyglass Dicer (a commercial visualization tool for Macintoshes).
8. IBM POWER Visualization Data Explorer.
9. Y0 (or YNOT), a spatial data analysis and visualization system being developed under a contract from Unidata by Macdonald Dettwiler.
10. Wavefront’s Data Visualizer product.
11. Mathematica.

This dependence on XDR is also one of the drawbacks of netCDF. XDR has a limited number of data types: byte, character, short integer, long integer, floating point and double precision floating point (storage size of 8, 8, 16, 32, 32 and 64 bits, respectively). Thus data must be expanded to fit the next largest data type. A worst case example would be 9 bit data, which would have to expand to 16 bits. NetCDF does not directly provide a method for data compaction, but one could be developed quite easily if the compressed data size is restricted to 8 or 16 bit data sizes. Ideally, this packing would be done by the library when the data was written by specifying the packing through either a global variable or a function parameter and transparently when the data was read (i.e., with no changes to the specification of the reading functions). It could also be done via a non-transparent mechanism (e.g. via attributes), but this would require that the software recognize the method used to pack the data. However, these compressions still may end up with a large expansion of data volumes (for example, if the data can compress to a 9 bit size, thereby still needing 16 bits to represent).

If arbitrary data sizes are required, this would involve either extending XDR to support new data types or scrapping the XDR layer for some other alternative. The first choice involves a great deal of machine dependent work (the reason the UPC chose XDR was to avoid having to do such a task), and there are no candidates for the latter choice that are as universally accepted and implemented as XDR. However, this problem of arbitrary data sizes is a problem with the implementation of netCDF, and is not a limitation of the specifications of the library routines themselves. The use of netCDF for large data sets with odd data sizes will require modification of the netCDF implementation to handle these data sizes in their original size.

Another drawback to netCDF is that it allows only one dimension to be an unrestricted size. Thus, for example, a netCDF that contains soundings of a variable number of stations would have to define a maximum number of significant levels. Thus for each station space would be reserved and used for that number of levels even if the number of levels was less than that. A similar problem would exists with a netCDF for hourly data that retained the raw report (or some portion of it), unless it was implemented as a single array containing all reports (a heap) with pointers from the individual station entries to the beginning of the report within the heap. This problem is an actual deficiency of the netCDF specification itself.
3.2 E-BUFR

E-BUFR (Enhanced BUFR) is an extension to the WMO standard GRIB and BUFR formats proposed by the STORM Project Office. Unlike netCDF, E-BUFR is a data format and not a library of data access routines. Fields are defined within each data record based on the position within the record. Records are stored in BUFR or GRIB format, and thus the data format allows for arbitrary data sizes and is very compact. The header record contains enough information to allow for some data sorts, merges and subset selections without decoding the enclosed GRIB or BUFR messages.

The advantages to E-BUFR are its ability to use variable sized data, its use of meta-data and its use of WMO standards (GRIB and BUFR) within the data records. Disadvantages to E-BUFR include its lack of a true self describing data format, the need for external tables to determine the meanings of the data fields, the inability to do arbitrary slices of the data and the lack of a software access library that works on a variety of systems. Basically, E-BUFR is another way to “package” raw data, especially data originally sent in GRIB and BUFR. Thus my comments below that mention GRIB and BUFR also apply to E-BUFR.

4. Method of Data Access

Another aspect of the question of data formats has to do with the way users plan to access and use the data. Your view depends on whether you see CDROMs as just a convenient way of distributing the data (i.e. publishing), or as a way also of providing an easy way to directly access the data. If the data is packed in a format like netCDF which allows easy access to arbitrary slices of the data, the user could be expected to use the data on the disk directly. If the data is packed in a format that does not allow such slice (e.g. GRIB or raw formats), a portion of the data would have to be run through a translator and staged to a read/write disk. Given the speed advantages of a magnetic disk over a CDROM, it may be desirable to stage the data in either case because of the faster access time. Requiring data staging will restrict use of the disk to systems that have sufficient magnetic disk to stage enough of the data for the application in question. The magnetic disk storage required could be considerable if taking a slice of the data different than the way the data was stored, for example a long time series of several points in the grid point data.

One approach is to distribute the data in several formats. The real cost of producing the disks is in the gathering and preparation of the data, the mastering costs are low in comparison. By producing the disks in both a netCDF and a “raw” format, we could let the “market” drive the ultimate decision. If the demand for netCDF disks ends up being low, we could decide in the future to not produce them. Similarly with the “raw” format disks.

The next sections will discuss each data type and the different formats available.

5. Grid Point Data

There are three choices formats for the grid point data: the method used at NCAR for the NMC gridded data (aka the infamous NMC octagonal grid), GRIB and netCDF. The first two are very similar in their method of packing the data. GRIB is more flexible than the NCAR format since it
is meant to represent a wide variety of grids, but software exists on more platforms for the NCAR software. GRIB is also currently going through a major revision. NetCDF would be the method of choice if the data compression problem was solved. For example, based on a total storage capacity of 680,000,000 bytes on a CD-ROM the height data from the aviation model run (10 levels, 73x73 grid, 10 forecast times) takes up approximately 820,000 bytes in GRIB format, 2,100,000 in netCDF format. Thus an archive of the initial and 6 hour forecasts would get 2,073 days on one disk in GRIB format, and 795 days on one disk in netCDF format. Possible compressions for the netCDF file include a pressure dependent compression to 16 bits (1,586 days on a disk) and a pressure-latitude dependent compression to 8 bits (3,154 days).

Note that satellite data is a special case of grid point data, and can thus be stored in similar ways to the grid point data.

6. Station Data - Synoptic and Upper Air

There are three obvious choices for this data type: raw (ASCII), BUFR and netCDF. BUFR and raw data would be difficult for taking arbitrary slices of the data (for example to plot a 500mb map from the soundings). Also, there are few decoders available for BUFR, so this is a difficult choice to make unless a significant software development was undertaken to support it. Since synoptic and upper air formats are well defined and fairly easy to decode, there would be no necessity to keep the raw data around along with the netCDF. The data expansion is not as great as with the grid point data. I ran several tests using the WXP version 4.3 decoder from Purdue to convert raw data into netCDF files. A raw upper air file of 138,684 bytes yielded a netCDF of 236,968 bytes. A raw synoptic file of 91,638 bytes produced a netCDF of 144,500 bytes. A netCDF files the same size or small than the raw data could be produced by compressing the floating point data into 16 bit integers.

7. Station Data - Aviation Hourly

There are the same three choices for this data type: raw, BUFR and netCDF. The problem with the aviation hourlies is the complexity of the observation coding. With the combination of errors in coding the reports by observers, possible errors in the decoders, and the existence of plain language remarks that cannot be decoded, some if not all of the raw data may need to be retained. A test with the WXP decoder on a raw file of 130,677 bytes yielded a netCDF file of 232,980 bytes. This netCDF file included the latitude, longitude and elevation of each station with 25 bytes allowed per report for remarks. By compressing the data into 8 and 16 bit integers (depending on data type), this netCDF could have be reduced to approximately 170,000 bytes. 35,950 bytes of this are used for the remarks, which are blank or less than 25 bytes in many cases. If a heap was used for the remarks it might save half of this.

There are other problems with using netCDF files with hourlies. AMOS stations report 3 times an hour. Other stations report specials off the hour. Some stations only report part of the day. The netCDF file would have to be carefully constructed to avoid a significant amount of wasted space. However, I feel it is important to have the decoded data on the disk, and netCDF seems to be the
best method for accessing such data.

The recommendation depends on the confidence one has of the correctness of the decoder and the need to keep the raw data available. Here are the options I see:

1. Just include a netCDF with the decoded data and pointers to a heap containing the portions of the reports that where not decoded (e.g. remarks). Thus is the decoder could not handle a report, it would be in the heap in its entirety. An archive of the raw data would be available for special requests.

2. Have both the decoded and raw data on the same disk.

3. Put the decoded and raw data onto separate disks.

8. The Facts of Life

So much for the technical questions. The reality of this discussion is that there is more than technical facts involved in the decision of which format to use. There are also the non-technical aspects regarding whether to change the format from its original format at all (assuming that the data volumes are similar). There are two extreme views that can be taken. The first is to not change the format at all, the second is to change everything to some sort of “common” data format. Which view you take in this discussion depends basically on whether it is best to change the software to match the data, or the data to match the software. Different people take opposite sides on this depending on:

1. Their opinion on which is less expensive, reformatting the data or developing the software.

2. Whether the person asked is responsible for the data or the software (i.e. the argument over “whose ox is gored”).

Some technical advise can be given to the first item above, but opinions differ greatly as to the relative costs of data preparation versus software development. The second is purely political, and cannot be answered in any technical sense whatsoever. The most likely outcome of this argument will be that both the data and software groups will try to minimize their costs. Thus, the data will be left in its original form, the software developers will support only one format, and it will be left to the end users to write software to reconcile the two, which will be a “hidden” cost of the data format decision. Hopefully, such an outcome can be avoided, but this will be the tendency of the different groups.

9. Conclusions and Recommendations

1. The data expansion problems with netCDF for some data types (i.e. grid point data) make it impractical in the short term. Work should be done in cooperation with the Unidata Program Center (UPC) to see if netCDF can be implemented for arbitrary data sizes and if data compression is possible. Until this is done, the grid point data should be produced in the GRIB (or E-
BUFR) format only. Since GRIB does everything (and more) that the NCAR data format does, I would recommend against using that format. If a storage efficient way of producing netCDFs becomes available, I would then recommend producing disks in netCDF also.

2. Produce separate disks of the upper air, synoptic and hourly data in raw and netCDF format if the netCDF version can be produced without too much data expansion. The netCDF format for the hourlies should include the portions of each report that were not decoded.

3. Review these recommendations after two years to see if the numbers of disks sold in each format justify its continuation.
Section 16

Information About Formats (Dec. 1991)
Information about Formats

Following are some considerations about formats. A little information about display routines is also included.

One of the questions we are trying to sort out has to do with making CD-Roms. Should we try to establish a common format for a CD-Rom? Or can we leave the CDs in good native formats and make any needed conversions to other formats at a later stage?

In this text we will make a number of comparisons between formats used by NCAR Data Support and those used by Net CDF.

1. Display packages used by HAO

HAO uses a couple of graphics systems which have a lot of displays that are useful for astronomy. One of them is more general than astronomy:

a. IRAF system for astronomy. Ray Bovet, NCAR, said that his system was built on the design philosophy of first defining “the one true format”.

b. IDL system. This system can handle any format. The user describes the format to the system. The system does not force exact formats onto data producers or onto other users. It includes routines for low and high-pass filters, etc. This started in astronomy, but it is not restricted to astronomy. It has many facilities to help lots of users.

Ray Bovet says that IDL is completely flexible to deal with any type of data or format. The user writes a “procedure” to write the data into an array, and then IDL provides many tools to work with the data graphically.

There is a manual for IDL, but it sure helps to have someone to answer questions when you are learning it. Bovet guesses that it takes a couple of days to really get started in using it. IDL is popular in HAO at NCAR. It is an environment that HAO scientists say they are happy to work in. Over a dozen people are using it. A problem is that it costs about $3000 per workstation.

c. Comparisons. The Net-CDF system is like the IRAF system in its design philosophy. It defines a standard format rather than adapting itself to other formats.

2. CD-Rom activities of NGDC

The National Geophysical Data Center (Boulder, CO) has been one of the leaders in the CD-Rom area. Their first one was produced in 1987, about the same time that Cliff Mass entered the game.

I have seen partial demos of their systems several times. I understand that they stick with native formats on the inside. I’ll try to find out what methods are used inside the system if I get any time. They have given me a standing invitation. I understand that the displays are prepared once and then they will work on DOS, Mac, and Unix systems.
3. Categories of Data that are displayed

The data that a user will want to display are in the following categories:

- Data from the user’s calculations
- Data that are in a “system” format
- Data in a stranger format.

If the “system” design in such that it requires that all data be in its own format (as for Net-CDF) then the options are to convert a lot of formats or to push at the data suppliers to convert their data to your own format. Most data suppliers will not convert their formats to something specified by a given user.

In any case, the user of a given system will want to display data that he/she has created or that are in a stranger format. I think that most systems will have happy or irritated users, depending on how easy it is for users to get data in and out.

Some systems are being designed that do not require that all data be in system format. In this case, the system can carry along a string (or blob) of bits that define a logical record (of variable length). The format can be produced by another group. The system knows how to operate on the blob of bits to extract the data it needs. This seems to me like a clean approach to the problem. The question is whether we have to give up performance to take this approach.

4. Datasets and Formats

We will consider the format question as applied to several categories of data. These categories are as follows:

- Selected large datasets
  - This includes global GAC (4Km, 5 channel) satellite data, TOVS sounder data, large sets of analyses from weather centers, etc.
- Large sets of observed data
- A lot of small datasets
- Data for special science projects
- Other data, sometimes mid-sized

4.1 Selected large datasets

This data category includes large sets of satellite data, large sets of analyses from Weather centers, large basic radar datasets, etc. It simply wouldn’t make any sense to put such data into formats like Net-CDF or CDF.

Consider the volume of some of these datasets:

- Global 4Km, 5 channel GAC has a volume of about 250 GB per year for 1.2 satellites.
- Global NOAA TOVS sounder data has a volume of about 500 GB for 6 years, for 1.8 satellites
- One GOES satellite produces about 3500 GB per year.
The Net-CDF format would blow up the volume (already large) of GAC and TOVS data by more than 60% (at best), because it is 10-bit data.

The increase in volume for basic GOES data would be about 25% because the visible data uses only 6 bits.

My impression is that these original formats are all simple scan line formats that are well thought out by the satellite instrument teams. The only reason to change them that I can see is to take advantage of data-compression methods. Rather simple compression methods can reduce the volume by a factor of 2 or 3, or more. The best procedure is to convert samples of such data to an end-use format such as Net-CDF only when it is necessary. This means that the data would not be converted in advance and that the basic archives would not be affected. It also means that the people who do calculations with large amounts of data would have less bulk, faster data rates, and less to learn.

4.2 Large sets of observed data

In Data Support, a considerable amount of our time goes to preparing very large datasets that have individual observations, often with global coverage and covering many years of time. The requirements for handling these large projects are:

- Very compact formats, but usually still simple
- Variable length capability
- Checksum control so there are no mechanical errors
- Absolute volume control (so that a human or hardware error can't lose data).
- Data flow methods that permit very fast data paths and fast sorts (that use the 7-way sort-merge methods).
- Control of the format so that all needed variables and flags can be included. In this process, we cannot afford the time to coordinate with 100 other groups. It takes a lot of coordination as it is, and the results would be hurt by letting all of the world's format questions get mixed up with the science.

What are some of these projects?

- COADS project: prepare the world's ship and buoy data, from 1854 - on
- World aircraft data: prepare data for 1960 - on for reanalysis
- World rawinsondes. Prepare data for 1948 - on. This includes data from GTS (telecom in real-time), as well as data from the archives of many countries
- Satellite cloud-drift winds. Prepare a dataset

Where does Net-CDF fit into these projects? It doesn't have the properties that are needed for projects like these. A conversion to Net-CDF could be made later, but it would blow up the volume a lot. The people who just want to use the data would be better off with the basic format (with access routines), rather than a format like NetCDF.
4.3 A lot of small datasets

We have many datasets such as methane from cattle, cloud statistics, ocean wind stress climatology, ocean mud cores, etc. Most of these are in simple character formats as defined by the scientific groups that originated the data.

Suppose that we converted all of these datasets to one of the "common" formats as some advocate. The use of most of these data is rather specialized, and is such that scientists wouldn't usually want to take advantage of the special displays offered by Unidata or other systems. A lot of time would be used to convert the data. It also looks to me like most PIs would spend a lot more time trying to get the data back out of the common format (eg Net-CDF) than they would to just use the basic format.

The situation would be different if Net-CDF had the design structure of being able to accept the original format, and wrap a format unpacker (and perhaps a data descriptor) around it. Then the same basic data structure could be fed into either display routines that work using Net-CDF, or it could be used in the other fast data streams.

4.4 Data for special science projects

NCAR Data Support started a project in September 1987 to prepare climate model data to use for assessment studies. In these studies, the effects of climate change is modeled for such sectors as forests, crop yields, river flow, and economics. The climate model data is from a number of the world's main model runs. The main assessment studies have involved about 40 PI groups in the U.S. and 45 overseas. Probably about 50 other groups have also obtained the data.

Our job at NCAR has been to make it extremely easy for these groups to use the data. We obtained data in many formats and put the necessary subset of data into a simple character format. It was very easy for the users to insert the data into their crop or river models, usually run on PC computers. It would not have been easy, useful, or appropriate to force all of these users to learn the Net-CDF systems. It is also true that people that do use Net-CDF could find this data very useful. A translator at the time of use could fill this need.

4.5 Other datasets, sometimes mid-sized

There are various other datasets that are used quite a lot. One of these is the world monthly surface data. Many people use it who would not want to be forced to learn Net-CDF or another system. Others would probably want to use the display capabilities of a system. This has been one of the most popular datasets in the NASA Goddard CDF-system. Most users now receive the world monthly data in a simple character format.

During the last 18 months I was in 2 meetings where several were extolling the virtues of SASS. In another meeting, people were all for Ingress. It should be easy to import data into any of these systems.

How are datasets used?

Figure 1 summarizes how datasets are used. The categories of use are:

1. Large batch computing jobs
2. Personal browse
3. System browse
Data Support (DSS) has concentrated on doing a good job with #1, a reasonable job with #2, and we make the data available for other groups to support #3.

Unidata concentrates on doing #3.

I have been in several meetings where scientists say: “please give us the data so that we can do our own thing.” This is often interpreted as meaning bigger and better display systems.

It often amuses me that people use the high data rates of EOS to justify big data systems, and then design data systems that are not appropriate for handling large volumes of data.

It is still true that there is a real need for graphics of many types. Some of this capability will be developed within our scientific community. Some will be commercial software. The user should be able to match up the data with software that he/she chooses.

6. Efficiency of Formats (Volume)

A good binary format typically requires only about 40% of the volume needed by the equivalent character format. For example, consider four digit decimal numbers that range from 0-4095. These numbers can be stored in 4 characters (32 bits) or in 12 binary bits. The character data has 2.67 times more volume. Net CDF is a binary format. The volume of data that it produces should be less than for a character format. We will see that this often is not the case.

A binary format that forces the boundaries of numbers on to byte boundaries (8, 16, or 32 bits) will typically blow up the volume of carefully packed binary data by a factor of 50%. The case is much worse when there are a lot of short 1 and 2-bit numbers such as in COADS ship data. My guess is that the volume of COADS ship data would be a factor of 3 or 4 larger in Net CDF. COADS is a large dataset; the large increase in volume is not acceptable.

Consider grid analyses. If the format can’t handle scaling and base values, the volume blowup will be much worse than the above case.
HOW ARE DATASETS USED

1. Large batch computing jobs
   - Data flow is important
   - Ease of use is critical

2. Personal browse
   - Use various graphics routines, publishing programs, etc. (eg. NCAR graphics)

3. System browse
   - Use data in image analysis routines
   - Unidata graphics
   - GIS systems, spreadsheets, etc.

Comments:

For 1: People should not have to learn a big system to use the data. System overhead should be very small.

For 2: Use data in a variety of routines

For 3: Not one big system. User should be able to feed data into a variety of routines. The user should be able to choose the routines.

Figure 1: The use of Datasets
If the format is limited to byte boundaries, the volume blowup of much satellite data will be very large.

- AVHRR and TOVS have 10-bit data. These are large datasets. The minimum blowup of the volume would be 60%.

Data compression. If a format structure does not permit selected types of data compression the data volume blowup can be very large. If a format forces fixed length on all of the data reports in a file, there will be an additional blowup of the volume.

Net CDF:

- Is organized as an 8, 16, and/or 32 bit system.
- Does not handle scaling and base values of grids.

WMO formats:

The new WMO formats, Grib and Bufr, do not always force data onto byte boundaries. Net-CDF does.

7. How does Net CDF handle some formats?

- Raobs: Now it uses a maximum number of levels, and just wastes the space for shorter reports. This gives the fixed length reports that are needed.

- Grib analyses: These are put into 32-bit floating point numbers. This is apparently why Edmond's volume increased so much.

- Hourly reports with remarks: Russ Rew thinks that these are in fixed length reports, where each has a pointer to the remarks. In a given report the pointers might say that remarks start in byte 2490 of a separate buffer and are 47 bytes long. The separate buffer has a fixed chosen length to hold remarks from all reports. What if the weather is very messy on a given hour, and the remarks buffer fills up? Russ Rew thinks that additional remarks would be lost.

- Net CDF length for hourly data, a question. Usually a binary format has less than half of the volume of the equivalent character (GTS) data. Why does the Net CDF data have more volume than the character version in Harry Edmond's tests? Russ Rew says that Net CDF puts some of the variables in 32-bit floating form, ready for the applications.

- Would it be possible to implement a truly variable length structure in Net CDF? Russ thinks that someone could probably implement a variable length layer on top of Net CDF. It would make things harder, and the interface would be messier. Things like slab access routines might be hard to implement.

Rawinsondes data

When we got raob data from Asheville in the late 1970s, it was in a fixed length format, with a maximum size of 80 levels for a report. They finally helped us by developing a capability to put the soundings out in a variable length format. This helped a lot; there was a difference of a factor of 4.5 in the data volume.
The US NCDC raobs have a lot of levels, an average of about 55 per sounding during the 1980s. A few soundings have over 127 levels. This seems excessive.

The NCAR archives of rawinsondes data would blow up by a factor of about 10 if we used fixed length blocking. This would be ridiculous.

**Keep data packed on slow devices**

When packed data are used for calculations or displays, it is best to leave them packed up to the point of use. Tape channels and communications are a lot slower than main memory speeds. It is really bad practice to blow up the data volume and then send the extra volume over communications nets.

**Time to write an array of Cray numbers**

Frank Bryan, NCAR Ocean Group, had an array of numbers (100 x 100 x 30) to write from Cray memory to Cray disk. In native Cray format, the time needed was 1.0. If flexible File Format was used, (a routine by Cray), the time needed was 2.0. In this routine, the 64 bit Cray words were output as 32 bit IEEE words. He output the same data using NetCDF, also as IEEE words. This took much longer - between 100 and 200 time units. Russ Rew looked at this problem. Apparently the Cray makes a slow function call for every word.

**Planetary Image data at JPL**

JPL is producing several hundreds of CD-Roms that have planetary images. The image data is already rather tightly packed, but JPL is going further. Their CD-Rom projects are practical within constrained budgets because they are compressing the data so that it requires only 28 to 33% of the original volume. This is described in more detail on Page 15 of:

Jenne, Roy L, 1991; Data available on CD-Roms, NCAR, Boulder, CO

When a PI compresses data, we end up with variable length data reports. In the data business, we should always have data structures that are flexible in permitting variable length packing. During the past few years, we see too many data structures that do not have a good design in this respect.

8. **What would a good data structure look like?**

I can think of many times in the past 30 years when format questions have come up. Many people have considered what constitutes a good format versus a poor format for given types of data. Some groups have defined common formats that are supposed to solve all of the problems. About ten of these are in the “Data Provided by NCAR Data Support” section.

During the past few years, I have run into more people who question the idea that one set of format constructions is best for everything.

I think that good data structures start with the idea that data is made up of a series of logical records that may be of either fixed length or variable length.

- We need a few better “packaging” standards that carry this variable length data. The packaging should keep the data under checksum protection.
• A logical record might be a satellite scan line, a grid point analysis field, or a rawinsonde report. The data could be unpacked either by using the fast "data access routines" that the Data Support Section now employs or by going through the more detailed access methods needed for interactive displays. The display routines would either decode the format or unpack elements from the existing structure. In the latter case, the data could be used by data display packages.

• Indexing methods. If reports are of equal length the indexing methods are fairly simple. Otherwise an array of indices would have to be kept. If one usually has to access a CD-Rom to find an index, a lot of extra time will be used.

In Nov 91, J.T. Young, University of Wisconsin called to ask about ideas on formats. I hope that we can all share a few more ideas without using a lot of time.

9. As easy as ASCII

This is about a plea for simple ASCII formats. I am a member of the Users Advisory Group for the Goddard DAAC. The first meeting was December 2-3, 1991. One of the committee members is Al Arking who was the head of the Goddard radiation branch for a number of years. He made a strong pitch for the idea that users just wanted easy access to the data and would use their own display routines. He preferred to get the data in an ASCII format, "that anyone can easily use." I do not think that character formats are appropriate for all data, but I think that the standard for ease of use should be "as easy as ASCII." I should also say that there are some poor character formats that are not easy.

10. Claims for common formats

There are at least 10 common formats that I can think of. These are listed in Appendix 2. The list gradually changes with time. The time scale of many changes seems to be 3 to 5 years. I like to see systems that can adapt themselves to using a few different formats, rather than converting everything whenever a new idea comes along.

11. Forcing changes on users

DSS has many users who already have software that includes DSS data access routines. These routines are very fast, they are based on existing efficient formats and they are usually very easy for users to install. There are many users who prefer this way of working with data because they do not have to learn another system. These methods provide unpacked data ready for calculations or for "personal browse".

We do not want to force these people to make changes, especially changes that are harder to implement, that increase the data volume, and that increase the running time of programs.

12. What is happening at NMC? Gempak

Mary Des Jardins wrote Gempak to display meteorological data. It is one of the systems supported by Unidata, and is quite popular with a number of universities. Gempak has its own file structure on the inside, but I think that it can now accept data from NetCDF.
Mary now works for NMC, to help solve their display problems. She is still deciding some of the ways to carry the data. In the case of grid data, the Grib format will be used to archive or carry data. In a few cases, she will have to fix up Grib a bit. When the data is being used, I believe that it goes into a different array structure.

13. US Interagency Standards Group

There is an Interagency working group for data management (NOAA, NASA, DOE, NSF, etc). It set up a committee to consider standards. Eliot Christian (USGS) is the chairman. I talked with him in Nov. 1991. Their report is not yet done. From the conversation, it was clear that they will not advocate just one format structure for the US.

14. Format ratings

Let’s look at a few “top of the head” ratings for the format approach taken by Data Support (DSS) compared to Net CDF. On this scale 0 is lowest and 10 is most.

<table>
<thead>
<tr>
<th></th>
<th>DSS Method</th>
<th>Net-CDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Simplicity for bulk data user</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>2. Simple for personal browse</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>3. Simple for Unidata browse</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>4. Simple for browse using other systems</td>
<td>2</td>
<td>2 or 7 (7 if x-late done)</td>
</tr>
<tr>
<td>5. Is the format compact</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>6. Handle 10-bit satellite data well</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>7. Handle compressed data well?</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>8. Handle ISCCP cloud data</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>9. Handle other formats for input</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>10. Handle formats at bit level?</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11. Handle scaling &amp; exponents?</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>12. Degree of format unification?</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>13. How close to future data methods?</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Of course, such a list is partly amusing because one can ask many types of questions.

15. Some comments and issues

1. Net-CDF is not adequate for an overall archive format.

2. Users have many program codes that already use DSS methods.

3. For a few projects, DSS tailors a format for the needs of the group (character model formats for EPA).

4. The US Interagency group will almost certainly adopt a philosophy that permits some format flexibility.

5. The DSS methods generally work well because we do not try to force data suppliers to change their formats.

6. It looks to me like the best of present and future data methods do (or will) use methods that permit a good measure of format flexibility.
7. If DSS puts out a lot of data in Net-CDF format, our workload will increase a large amount because we will normally have to maintain the present format also.

8. Our attention to these format questions is hurting our progress on the reanalysis project and other projects. We started out (June 1991) thinking we could handle both CD-Roms and reanalysis if we could keep methods simple.

9. Net-CDF depends more on error-free data paths, than methods that include some form of checksum protection. Trouble doesn't happen very often, but it does happen.

10. People should be able to use Unidata routines to view some of the data if they choose. This can be accomplished by:

   • Conversion to CDF in advance
   • Write CD Roms in native formats. Then Translate to Net CDF or other general formats when necessary.

16. How does Net CDF fit in?

   Dennis Joseph and I talked (Dec 9) with Russ Rew about various format and packing issues. Russ said that the main purpose of Net CDF was to be at the end of the data chain and handle the data files for display routines and other applications. Since it increases the data volume for a number of cases, it probably isn't the ideal form to use to transmit data or to store data. Russ felt that Bufr and Grib were probably better for those functions.

17. NASA data systems

   The process of defining future NASA data systems is still going on. The seven DAACs (distributed archive centers) have been established. NASA has said that native formats would be supported. They will work for a year to figure out whether to use a "common" format also - such as HDF (seems top in thinking), Net CDF, the JPL method, or CDF. I talked with members of the Goddard group on December 4. We agreed that a meeting was needed to sort out some of the issues.

Two texts give some information about the NASA history and debates:

   • How to Distribute Data, an Example (24 Feb 1989)

     The JPL NODS (ocean) system was a system set up where a user could get ocean data by communications. It cost $2 million per year and had little use. The relations with the ocean science group at JPL were not good. In 1989 it was restructured to cost less and to concentrate on data instead of systems.

   • NASA data systems (6 Dec 1991)

     This text shows some of the 1991 thinking going on with Goddard scientists. There has been a fear of EOS-Dis "big expensive system" planning for several years.
18. Data should be outside of systems

Data structures, in a real sense, should be outside of the systems that operate on them. The data structures store the data, and move it from place to place. I don’t like to see data systems that “own the data” as tightly as a DBMS systems. In that case the data can be lost when the system changes. DBMS systems can be useful for a number of tasks. In that case there should also be a copy of the data in a straight-forward external format.

I think that Net CDF largely meets the requirement for keeping the data external to any given system.

19. Unified parameter codes in formats (the problems)

If a given automated display system handles data from many sources, how does it keep track of the codes for wind, or temperature, or biomass in the many data sets?

Suppose that a given air temperature grid in one data file has a code number of “5”, but other air temperature data has a code of “7”. If a display system needs to recognize both of these grids as “air temperature”, we need a method to communicate that fact. Do we need to change all of the world’s data so that it has the same code? Is there a better way?

First we will consider the example of grid data formats. Organizations like NMC and ECMWF have many gridded fields that they produce, both analyses and forecasts. A format is defined that includes all of these grids in one structure, where each field is labelled by date, hour of forecast, type of data, level in the atmosphere, model used, etc. Actually this is the ideal; at NCAR we still get a number of NMC outputs in non-standard formats for special cases.

Consider the international Grib format for gridded fields; several centers have defined this new code that includes a few format options. We have already obtained many tapes from ECMWF that have “internal ECMWF” identifiers rather than Grib identifier. Each center has its own history so that is is probably hard to revise everything to look like a new format such as Grib. Also there are a few rare people who prefer to get some work done rather than waiting two years for official agreement on a new code identifier for a new field. A local code will therefore be assigned.

Some groups lobby hard for complete standardization of formats across all of meteorology plus other related disciplines as well. This causes several problems. One is that the format and the system becomes more important than whether the format is sensible for the data. Another problem is that a real bureaucratic nightmare is created by the problem of having to keep descriptors consistent across many types of data. For various reasons, the cost of this approach seems to become high.

As a person coordinates with more and more groups of people, the time for coordination probably increases with $n^2$. Also, the probability of any one group being happy with the resulting format is about $1/n$. Since these groups will be attending many meetings, it will be assumed that the importance of this work is represented by $n^3$. This all leads to a format history that includes official formats (often with limited use), format wars and the working formats. The working formats are often very good.

What should we do for applications that require a common set of descriptor codes? We can invent mostly automated ways of handling this at the time of data ingest. When it
can’t be fully automated, we could make it easy for a user to tell the system what code mapping should be used.

20. The Fortran list statements, and data variables

The question is whether we need to get everyone to adopt a common set of data descriptor codes for all data. I think that the answer is “No.” We will use Fortran list statements as examples of how to pass data or data descriptors.

One of the beauties of the Fortran subroutine call is that variables inside the subroutine can have a different name than they do in the user’s program. We usually do not think about this fact anymore, but it produces an enormous simplification. The user does not have to worry about what a variable was called in other routines. I believe that some data system planners are now falling into the trap of demanding too much planned commonality when there are better ways to conceive of systems.

There is a similar correspondence of data names and actual data correspondence values on Fortran IO statements with a data list. Each user can choose his/her own set of names for the same data in a record.
Section 17

Use NetCDF on CD-ROMs?? (Dec. 1991)
Use Net-CDF on CD-Roms??

This is a brief summary of ideas aimed at the question of whether to build CD-Roms in Net-CDF format. See the main text also.

1. When Main data use is Net-CDF displays:
   - There are some good displays based on Net-CDF
   - Net-CDF gives graphics developers some defined general formats to work with
   - If this is the primary use of the CD-Rom, this format has an advantage

2. When Data are used for calculations and personal browse
   - Timing and space penalties would be imposed by Net-CDF
   - Users would all be forced to learn this system.
   - It gives some added capabilities for personal browse, but users would still have to learn it.
   - Many users would prefer simpler data delivery methods than Net-CDF
   - Compared with using good access programs, people whose main motive is to make calculations, would be frustrated.

3. Net-CDF still lacks properties needed for some tasks
   - Uses byte boundaries, not bits, within a report
   - Blows up size of many formats
   - No smooth variable length capability
   - Can’t nicely handle various data compression options
   - Seems excessively complex and “unclean”
   - Does not have checksum control on blocks
     • Checksum protection of archive data is very desirable.
   - Is more like an internal working level format than an archive or a user format.

4. Access programs
   - Are very fast and straight-forward
   - Are easier than Net-CDF
   - Most users only deal with 1-4 datasets in a given period, so the access programs are not a big bother.
   - When a user (or a display system) has to deal with many datasets, the integration problem has to be faced.

5. Methods that permit some format differences, but are still integrated
   - NGDC leaves the data native, but describes format to system
   - Glen Flierl advocates this approach (he does ocean work)
   - Sounds as if JPL uses some of this approach.
   - See the text for the IDL display system

6. When Net-CDF is not the CD-Rom format
   - Can still insert the data into Net-CDF
   - Have to select necessary data onto disk before cross-cut slices are made
   - There are about 200 CD-Roms, none in Net-CDF. People are making good use of the data.

Roy Jenne
17 Dec 1991
Section 18

Levels of Data Support  (Oct. 1988)
LEVELS OF DATA SUPPORT

We will consider different types of support that are offered to users of data. Depending on the main data needs to accomplish different scientific or applied goals, we often have to shift our priorities on what computing functions to develop. For given amounts of resources, we often have to decide whether it is more important to make more of the basic data available (and clean up its problems), or to provide more ways to display the data. Different levels of support for data use are:

1. The User Has Access to a Dataset on Tape

   - The user obtains the basic data on tape or other media;
   - Format information is given; other help is limited;
   - No access program to unpack data;
   - Perhaps wait hours for tape to be mounted;
   - Even to provide this amount of support, a considerable effort needs to be made to gather the data and organize it into reasonable datasets.

2. Dataset on Tape or Mass Storage System:

   This is like #1, only an access program is sent with the data. The user also has more variety of software, such as general graphics routines, to use.

   - Good format and other info;
   - Have an access program to unpack arrays for computing;
   - Have a selection of good inventories;
   - Data can be on-line within 2 to 4 minutes;
   - User computes with the data;
   - User has general display packages and other tools to use (data blocking, bit manipulation, etc);
   - User has many math and statistical routines to use;
   - Some data-specific software routines are available:
     - code to make common grid transformations;
     - code for common display routines; These may even include complicated routines such as: take weather data from 6 arrays (one for each hr) and show me a movie of how winds and weather have changed in the last 6 hours.
   - Project to gather display software;

   Many universities and laboratories develop selected display routines. It may be useful to have a small activity that gathers together these packages so that they can be generally used. Usually a laboratory gradually prepares
display functions for a given type of data. A modest proposal to NSF from a lab., to gather such software and make it available to a larger community, often should be supported.

Advantages: The user has full flexibility to do anything that he/she wants. He is able to have plots the way he wants them, not just accept what a system offers. He can also use "canned" displays when they are available. Most science will be accomplished in this mode. But, the user has to know how to program.

3. Like #2, Only the User is Remote From the Archive
- For complicated formats, a portable access program is made available;
- The existence of selected software is made known for data transformations, etc;
- User receives data on tape, cartridge, CDrom, etc; Small datasets or selected subsets may be obtained via communications.

4. Interactive Look at the Data, Using a System:

There are some well-defined cases for which many users need about the same type of display products. Also, instead of calling a subroutine, the user may just want to "hit a button" to cause the following to happen:
- Show me the present surface pressure map for my region;
- Put a dashed 500 mb map on top of the surface pressure;
- Show me hourly satellite pictures in motion for the last 24 hours;
- Plot a upper-air rawinsonde sounding;
- Draw an upper-air cross section through the reports from five cities that I specify;
- Show me the plot of 24-hour rainfall on top of the present 700 mb map; use different colors;
- Show me plotted surface winds and type of weather for stations in my area. Do the same, but show me each hour for the last 8 hours in a motion sequence.

For other types of data, examples might be:
- Show me energy use by the US, UK, Germany and Japan, plotted on the same time scale;
- Display a plot of geographic structures through the Appalachian mountains;
- Display fish catches by continents, for the past 40 years;
- Display the depth of the ocean.

These systems are very powerful and useful when the goals are rather well focused. They also tend to be expensive to create. The Univ. of Wisconsin has been a pioneer in developing such systems, with their Meldas system which is
being sold worldwide.

The systems also may be very limited. A user sees a display and says "That is very nice, but I would really like to see ... (another related display)." But, usually the system organization doesn’t permit the user to use basic computer data and graphics routines to create his own displays. Thus, the user gives up, uses the methods of #2 above or tries to talk the systems people into adding his display.

5. A Workstation Environment that Permits Some Mix of #2 and #4

- The user can use programming and software tools to create his own displays. Thus, he is not restricted by the limitations of the system.
- If some canned displays (as in #4) have been developed, a person can use them by "hitting a button."
- A PC has programs to do spreadsheet analysis, word processing, play games, etc. It would be desirable to have similar options to make data displays:
  — load a program to do a certain type of weather display;
  — use a program to show paleoclimate changes over the last 200,000 years;
  — a translation routine often will be needed to take the archive data and put it into a structure needed by a given display system.
- Inputs will be tape, CDrom, communications, etc.

6. Discussion

The approach used should take account of how much a given type of data will be used, whether there is a common set of displays, and what the costs are. Many sets are rarely used; for these, level #1 (probably adding an access routine) is what makes sense.

The canned system approach (#4) makes sense only when the objectives can be focused rather well and where the displays are used a lot. This happens in McIdas. It also happens in the types of weather displays and overlays needed in a forecast center. It happens for public TV weather forecasts.

The environment noted by #5 is the one that we hope will slowly evolve.

One should not try to force all archive formats to be the same as the structure that a particular display system wants. We don’t want to restructure all archive data every time a new display-level idea comes along. There is no necessity to change the archive formats in order to achieve these goals of data access.
Section 19

Data Formats and Methods of Data Access
(Mar. 1992)
Data Formats and Methods of Data Access

In this paper we provide information that we hope will help in the understanding of data format issues and data access issues.

Nature of the Format Discussion

Most scientists obtain access to data by directly reading datasets and selecting the data that they need. They often use “access programs” that make the process very simple. Software systems people enter into the picture. They say “We can streamline the process of data access. Convert all of the data to our format and we will use this format to deliver the data to users and to deliver it to display systems.” The PIs say they want a good data system, but one that is very easy to use and that is efficient. The systems people say they can provide a system that gives users all sorts of cuts through the data, and provides beautiful displays. The PIs say:

Sure, we like displays, but we want to choose our own displays. We mainly want access to the data in the simplest way possible, then we will choose our own software. We don’t want a system that is overly expensive and then does not even give us what we want...we have seen systems like that. When I just needed to read some data from a tape, I was unable to get any help. People only seem to want to work on their system.

The “common format” (CF) people have a strong shared view: All data should be put into one common format. And each group also thinks that data should be put into their common format. These groups say “What is so great about native formats? Why can’t other people see the advantages of one common format?” CF people think that if they only understood Freud a little better, they could understand the other points of view. Then the CF people and the direct access people drift together. They finally note that they both use flat file methods to store large datasets. At this point, a person enters the group who has developed airline reservation systems and taught computer science for 20 years. He hears the term “flat files.” Can people be talking about a modern data system and still be talking about flat files? He makes a few cautious remarks and leaves, since there is no reason to risk being massacred by this primitive group. He cannot wait to tell his friends about what he just saw.

Then the hardware systems group ambles in. “We hear that you have a few bits of data that you need us to store, a whole mountain, like $10^{17}$ bits.” The systems group says that they have systems designs that will do it if they can only get ten times the present channel speed and six times the amount of data on each storage cartridge. The others say “can’t you just use more channels if that costs less?” The systems group says “Look, we need a first-class system, fully automated. You do not fly piper cubs across the ocean. This is a lot of data. We don’t ever want a person to hand-mount a tape.” But what if we can save $20 million by hand mounting tapes that are only used once every five years. And people can mount the tape in only three minutes, compared to two minutes by automation. “Look, that is an old-fashioned design, we are talking about the new world. Isn’t it important to support global change research?”
If an agency only listens to the software systems people, they will tend to get a data system that is focused on the display of small amounts of data, rather than serving the broad data needs of the users. The system will also be much more complicated than a system that is focused on providing direct access to data, and to keeping learning time low.

In the broad context, users need both direct access to data, and they need to be able to use various graphics packages and data manipulation tools. Typically, the graphics and other software tools should be developed in a competitive environment, and then scientists should be free to choose the tools that they like the best.

How are organizations handling the format discussions? In Feb. 1992, I heard that NASA might choose HDF as the one “common” format that they would support for EOSDIS. In the fall of 1991, the world operational weather and ocean centers started implementing the WMO standard formats: Bufr and Grib. A display system has been developed in Monterey (Neons) that is based on the WMO standards. If HDF is selected in NASA, what will groups using other formats (such as CDF) do? Judging by preliminary discussion, they would probably continue to use their format but would also use JPL methods to describe and convert formats if they needed to send (or receive) data in HDF. In a few years, we may convert data 50 times before it trickles down to the user! This will be known as the system “trickles down” effect for data users.

1. Goals That Involve Data Formats

The way that we handle formats influences a number of things for users and for data systems. We need to find ways to handle formats that have several attributes:

- Gives users direct access to data with very low learning time, and does not require large software packages.
- Handles large volumes of data efficiently.
- Permits the invention of new ways to compress large volumes of data.
- Permits a reasonable amount of flexibility for the groups that prepare data.
- Allows several types of data to be imported into data manipulation and display packages without a lot of work.

What is happening can lead to a different thought: If the data already has quite a good format for a logical record, can’t we invent ways to use that format and still achieve the benefits of common formats? Would this be efficient?

2. Some Styles of Data Access

We will describe methods of data access that use “ordinary” formats along with access software. Access methods that employ one of the “common” formats will also be briefly covered.
2.1 Fast Input of Selected Data Streams

Users often need access to data to make many calculations or to browse some numbers. We will now describe the methods that are commonly used for this type of data access.

In these methods, a file of data is input into a user program. This file contains the data that the user wants and probably also has some extra data that will be filtered out during input. The user’s program obtains one logical record after another. The data flows through the program, and the user selects the portion of data that is needed. One logical record might contain:

- A gridded global analysis of winds in the mid-atmosphere
- A daily record of temperature and precipitation at Omaha for a one-year period
- Data from an oceanographic instrument that measures temperature and salinity as a function of depth
- A compressed image of a picture of Mars

How does the user obtain the actual numbers in these logical records? An access program is provided that unpacks the numbers and provides the user with an array of numbers that is the common starting point of any calculations. Perhaps the access program named these numbers in a way that does not match the names already being used in the user’s program. In that case, the user can take a few minutes to adjust the names to be what he or she wants. There is very little for the user to learn in this approach to data access.

If there is some data that the user does not want, the user’s program just ignores it, or probably does not even unpack it.

This approach keeps the data flows very fast and simple. If there are two different formats, those differences are handled by the access programs. For a few large datasets, this approach, without other tools, could cause the user to have to receive too much extra data. To handle these cases, a data selection is made for the user that eliminates most of the data that is not needed.

One advantage of these procedures is that NCAR obtains datasets from many PIs and many organizations around the world. We accept whatever format they use for the data. About 80% of the time we do not have to change the original formats, in order to make life simple for users. These methods handle either binary or character formats.

At the present state of development, these methods work well. Handling different formats is not a big problem. The main remaining problem is that the “data packaging structures” (ways that the logical records are grouped together) vary too much. Also, some data blocks are still not under checksum control. We are also finding that some of the new computer operating systems are not as smart as previous versions. In this case, we find that we have to adjust data structures to feed these operating systems baby food rather than ham and eggs.
These data methods allow one to incorporate many of the new ideas in formats as they happen in different subdisciplines. For example, if someone defines a new way to compress a GOES satellite image, it can be easily used. I do not think that any of the present “common formats” have this attribute.

What are these formats like? Some are just a very simple character structure output from a PC. This is a structure that anyone can quickly use. Some of these formats bring together all of the weather data structures that a weather center has worked with over 20 years. This gives a consistent overall structure for the center’s data, even though the structure within each type of report is different.

When are these methods most effective? They are popular for most calculations because the data flows are fast and there is almost nothing to learn. They also give what is needed for 50-year archive formats.

As the reader can probably detect, this is the main type of data access that we concentrate on in the Data Support Section at NCAR. The main goals are:

- Support users who want to make calculations using data.
- Support users who want to do “personal browse.” This means that they look at some data in their own program to “see” it. In the process they will probably display portions of the data. The user will choose the type of software that they will use to display portions of the data. The software will come from a variety of sources.
- The data can be input into various larger data display and manipulation systems. This can be done, but we do not give this highest priority.

2.2 Common Data Formats

There are actually many “common formats” in which different subdisciplines have defined data structures that they think meet their needs in the best way. During the last few years, when the term “common format” has been used in the fields of meteorology or oceanography, it has often been taken to mean one of the following:

- The WMO Bufr and Grib formats were defined about 1989-1990 and are now being implemented by operational meteorological and oceanographic centers.
- GF-3. This was defined in the early 1980s for use in distributing some of the oceanographic data. It is an IOC format.
- The CDF format was developed by Goddard about 1980-1984, revised about 1988, and then had a major revision about 1991 so that it can run on many computers, not just VAXes.
- The Net CDF format. This was the result of an effort at UCAR about 1985-1990. The format has been used mainly to hold real-time weather data, so that it can be fed into display systems at universities. The group used the ideas of the CDF format at
Goddard, but put in the hooks to use the data on several different computers, and to use network capability.

- The HDF (Hierarchical data format), developed at the University of Illinois super-computing center.

People who defined the common formats noted that most data can be considered as single numbers or arrays, and then designed the system.

All of the above data structures have the merit that logical entities such as an array of data, or all data from a rawinsonde balloon observation of the atmosphere is kept together. This is an advantage that often is not available in data systems (formal DBMS systems) where the data is often indexed deeply, down to each number. Such deep indexing does not work with large datasets. The DBMS approach may still work for datasets with fairly large volume, if the data are not indexed too deeply.

NASA has had teams compare the different common formats in 1991. But I think that they only considered CDF, Net CDF, and HDF. If a choice between these three formats were now (Mar. 1992) made in NASA, it would probably be HDF.

There are still problems in some of the “common” formats that I have looked at:

- Mainly they are built-on byte boundaries so that much of the high-volume data would be inefficiently stored.

- They lack variable length capability; therefore, the volume of much observed and analyzed data would be increased significantly. This also means that compressed data cannot readily be handled. This is somewhat too strong a statement, and a few caveats are needed.

- Lack of error detection capability (checksum control).

- The fact that they do not have some of the capabilities of Grib and Bufr (WMO code) is a problem. It is also the case that they have capabilities that are not in Grib and Bufr.

A Big System: The way that the data is actually held in these formats becomes complex. In order to get data in or out of the data structures, the user has to install a lot of software and learn quite a few rules. This often has the effect of making data use harder for the people who just want to use some data. For reasons of simplicity and the efficiency of data flows, many users will choose the methods that do not use the common formats.

2.3 An Example of Data Access for Climate Model Data

We will give an example of the use of climate model data to show two methods of providing access to data.
In the mid 1980s, it became clear that it was critical to obtain more quantitative information about the effects of climate change on crops, rivers, forests, etc. In 1987 EPA asked the NCAR Data Support Section to set up a database with the output from various world climate models that could be used for these studies. From 1987-1992, approximately 100 PI groups in the U.S. and 50 overseas have used the data.

It was interesting to interact with these groups. They are using very sophisticated models of the ecosystems and economic systems. The climate model data is a necessary part of their model runs, yet it should only be 5% of the overall problem that they have to worry about. These groups were usually using campus central computers in the early 1980s. By 1987, more than 95% of them were running the models on PcATs.

What Style of Data Support Was Used?

Since the part of climate model output necessary for these groups to use was small in volume, we chose a simple character format that would be very easy for all of these PI groups to use, and often gave them the data on a few floppy disks. Access routines were provided that made it easy to extract data subsets for their corner of the world, or even to obtain data for single and double CO₂ for every month at one grid point. This was very easy for the PIs and has worked very well.

Another Style of Data Support:

Some software systems groups want to force a different solution to this type of problem. The methods go like this: Put the data into one of the standard formats being advocated today (the formats have changed from what was advised a few years ago). Force the PI to load a lot of software into his/her PC to extract the necessary data. Learn how to use the software, then extract the data. At this point, the PI is back to where he/she was after 10 minutes of effort in the approach that was actually used. But not quite. It would have been unlikely that the data extraction would be as tailored to the needs of these PIs.

What Will the NASA Style of Data Access Be?

I think that it is still under debate in NASA, but I believe that they are groping toward some combination of both approaches, depending on the situation.

3. Systems for Display and Data Manipulation

There are many of these systems. Some of the main ones in meteorology and oceanography now are:

- McIdas system: developed by the University of Wisconsin
- Gempak system: developed in a research group at NASA about 1986.
• Neons: Developed by Navy and NOAA groups in Monterey, Calif., about 1991. Also used at INO in Mississippi and elsewhere. Focuses on the display of ocean data, and will be used on Navy ships.

• GrADS Developed in 1990-1991 at the University of Maryland. It focuses on data manipulation and displays for meteorology, with special attention to the needs of modeling.

What data formats are used in these systems? All of them have their own internal formats. However, Neons uses the WMO formats Bufr and Grib for the internal formats.

Users of these systems would like to be able to use a variety of different types of data. The design of a system should make it fairly simple for a user to import data from any format. The broader community needs to develop good methods to make import/export easy on a larger scale.

4. Data Access that is Independent of Underlying Formats

At the Jan. 1992 meeting of the American Meteorological Society in Atlanta, a paper was presented that was called “common formats; common problems.”

The authors noted that developers of data display and access systems often have two possible approaches:

• Put all data into standard formats
• Use smart software that interfaces between the archive format and the software

Several groups including JPL, NGDC (Boulder), and the Ocean GOES program are busy doing the research necessary to implement the “smart software” option. Some of this work has to progress further before all the problems are worked out, but it looks very promising to me.

I think that good methods can be developed to move data around between applications. This could avoid the “format wars” and other problems of the approach that demand a conversion to one of the several common formats.

One page of a proposal from NGDC briefly describes their freeform data access system, and it is attached.

5. Some Format Politics

For FGGE it was possible to define a set of formats that were easy, and that worked together in a uniform way. There were a few things that should have been changed, but it worked well. There was conversation at the time that these formats could become the common interchange format between world archives. This idea died. It was often easier to just accept the data in the format of the originating country. But for many data collection programs, you do need a common format for one type of data because there are a lot of countries.
In several cases, a data group has tried to force a complicated format onto a group of PIs. In this case there is often a rebellion, and the civil war slows progress. Finally the science group often obtains the right to prepare the data in their own way and the other group can convert it if necessary. Fortunately, formats are often negotiated in a reasonable way, and everyone remains happy.

I think that the following can help lead to good solutions:

- Create some better standards for data envelope structures.
- Provide some freedom to how groups define their own logical records (these records may contain analysis arrays, rawinsondes, satellite scan lines, etc.).
- Define better ways to describe the format of a logical record. This description can then be used to automatically handle formats when necessary.
- Make sure that blocks of data have checksum protection.

6. Examples of Large Datasets

With large datasets, it is important to keep the data flow fast and to keep the volume compact.

There are a number of large datasets. Some examples are:

- Datasets from selected satellite sensors such as SAR, GOES, Landsat.
- Sets of analyses from major weather centers.
- Basic weather radar data.
- Large datasets of conventional observed data. However, the satellite datasets are usually much larger.

A Note About Radar Data: In 1991, it was explained to several of us that the official format for basic scan weather radar data was a bulky format. In this structure, they apparently put too much header information onto each scan line, and needlessly increased the already high volume by a factor of 4.5. People asked if we should use the format because it was “official.” My answer was: “No, it makes no sense to take high-volume data and blow up the volume.” This would also blow up the costs and make it much harder for users to access the data.

7. How to Handle Large Datasets

There are some special data management requirements for large datasets. Some of these requirements are:

- Keep the format compact.
- Permit smart new ways to compress the volume, if these are also practical.
- Keep the data flow paths “clean” and fast.
- Give the user tools to rapidly unpack the data.
• Do not insert a lot of complicated software in the middle of the data flow path.
• If a smaller subset is selected, then the concerns about added software, or increased volume on local computers are not valid.

In some data designs, the volume of a large dataset is blown up markedly by putting it into a common format before it is put onto slow devices—such as a network or a CD-ROM. It is usually better to send it to the destination and then unpack it. Would this cause a problem at the destination? Probably not, if good access programs and translators are available.

8. **Prisoner of a Data System**

Suppose that a given data manipulation and display system has many good display features, but it is relatively difficult to put data into the system. What are the consequences?

A user will have data in other formats, but if it is hard to enter this data into the system, the user will be frustrated, and the utility of the display system will be compromised.

If data ingest is difficult, then the tasks will usually have to be accomplished by data professionals, but that will create delays and increase costs.

In the 1980s, there were complaints from NASA scientists that some of the NASA data systems did not have enough different datasets to use, either in the archive or in the display system. At that time, the Data Centers told headquarters that they would be glad to add datasets, but the cost would be $100,000 to $200,000 per dataset. Review panels noted that these costs were not sensible. It looks to me like many of these problems have now been alleviated.

We now know that there are ways to simplify data ingest. We need to continue to make progress in this area, and we need to study the amount of time being spent by users to access data.

*Exchanging Data and Graphics*

This was the headline of a short news article in PC Sources, Apr 1992, p.495. The first two paragraphs said:

No program is an island. If a presentation graphics program acts like one, then you’re stuck with the job of ferrying data and graphics back and forth to an isolated application.

A graphics presentation program’s ability to import and export files should be a key factor in your buying decision. The good news is that nearly all the programs profiled have competent import and export capabilities.
9. The Local Working Environment of a User

A user will have a PC, workstation, or big computer to use, usually two or three different computers. The data that is used will come from:

- Both distant archives and local media such as CD-ROM, Exabyte and other tapes
- Communications
- Calculations made by the PI

All of this data must be available for use in calculations, to make displays, to put into word processors, or to insert into various software packages for data manipulation and display. The PI will choose this display software from several options.

To make this happen smoothly, one of the big requirements is that the complexity and learning time that the user sees must be very low.

- It has to be easy to insert data into display systems or into other format systems.
- It must be easy to use data from formats that are not common formats.
- If the user only wants access to numbers, a lot of complexity or software should not be necessary.

The Problem: Many of the things being done to “improve” data access are actually making life harder for the user.

10. Data Linking in the PC World

In articles about data linking in the PC world, we see statements such as:

- New Wave does more than just put a new “object” coat of paint on Windows. Its Object Management Facility (OMF) brings the dream of transparent data exchange to life.

- Object Linking and Embedding (OLE) capabilities of Windows 3.1 hit the market even in advance of Windows 3.1 itself, which will arrive in April 1992. OLE comes into its own in Windows version 3.1. OLE brings some of the advantages of integrated software to standalone applications.

I do not know enough about these methods to have a feeling for whether the ideas may be useful for our handling of scientific data. A couple of pages from PC articles are attached.

11. A Future View of Formats

It is desirable to have a view of formats that can lead to data systems that will work with some diversity of data inputs. This does not mean that we should not regularize some formats so that we at least put one type of data into the same format. We have seen that we are not likely to get complete agreement on formats even if we wanted to. We also have noted that many formats have one problem or another, so that they are not appropriate for all data. So what do we do?
Consider a logical record that is in a reasonable blocking structure. The data in this record may be in a binary or character format, compressed or uncompressed. A given PI, group of PIs, or data center probably defined the format. The users doing calculations or personal browse will usually just use this record as it is, by using small access programs that unpack the data into arrays, ready for calculations.

Suppose that we want to insert the data into a data display or manipulation package. We have noted that these display packages use a variety of internal formats. If the user chooses, he/she can insert the unpacked array of numbers into the display package. However, with a larger variety of data, we may want to automate the insertion of data into these display systems. This can be done by developing a common way to describe the format of a logical record. This will result in a block of bits that objectively describes the layout of the data. A translation program in a display routine will then use this block of bits to automate the ingestion of the data into its own format. This is rather similar to the way that some display systems now ingest data from a common format. Common formats also have little blocks of bits that describe how a particular dataset is laid out within the common format.

What if a user needs the data in a common format that is not directly connected with a display system? In this case, the common format will have a similar translator to automate the ingestion of the data that is presented in this way.

Perhaps the blocking of the logical records has not been standardized. In this case, the data description will have to be in two levels, one for the blocking (or "packaging") layer and another for the logical records, which we have discussed.

There are several advantages to this method of handling formats. For PIs who just want access to the data (without using a larger system), we have met their needs. We have not tried to force all data into a data structure even if it does not make sense. This method permits people to use formats appropriate for the data and the problem. Some large datasets can be significantly reduced in size by using compression methods. These methods permit compression and variable length records. One key gain is that the system would be flexible enough that it could smoothly incorporate future ideas that are good.

Consider the data that describes particular formats; how are these descriptors carried along? In the WMO Buf format and in at least some of the other "common" formats, they are put at the front of a file of data. Everything that follows can be decoded using that descriptor. This would not work if the file itself was a mixture of apples and oranges. In this case, each report needs to say whether it is an apple or an orange, so that the proper descriptor can be used. The descriptors can then be in a separate small file that is only logically attached to the data file.

12. Data Format Guidelines

When groups need to define a format for their data, it is useful for them to learn from past experiences in formatting data. To achieve this, it is helpful to have written material (text and examples) that describe some of the various formats that have been
used (an overview). An experienced data person can also be a helpful resource in this process.

Summary and Conclusions

We have discussed methods to provide direct access to data that may be in different formats. We have also considered some of the common formats. Procedures that retain some variety of formats and still give a uniform view of the data have been briefly shown. I believe that total data systems should be able to tolerate some diversity in formats and be able to incorporate some new ideas that are good. We need to continue to support direct access to data, but to develop more tools that simplify it still more. Procedures that automate methods to easily use data in different formats should be developed. In display systems, there needs to be an emphasis on ways to easily import data that is not all in the same structure.