The Alabama Supercomputer Center

User Manual

Sixth Edition

Alabama Supercomputer Center
686 Discovery Drive
Huntsville, Alabama
35806

Publication	Date	Description
1st Edition	February 1988	Original printing
Revision A	September 1988	Minor typographical and editorial corrections
2nd Edition	June 1990	Updates and modifications of 1st Edition
3rd Edition	June 1993	Complete rewrite
Revision A	October 1993	New procedures and locations
4th Edition	January 1994	Update for Cray SV1 and editorial corrections
5th Edition	March 1997	Updates, modifications, and new format
6th Edition	July 1999	Updates and modifications of 5th Edition

Overview

This manual is provided for the users of the Alabama Supercomputer Center (ASC) as the primary reference for use of the Alabama Research and Education Network (AREN) and the Alabama Supercomputer Center facilities. The manual covers the supercomputer configuration, the physical structure of the network, available software and hardware, access methods, and user support.

Suggestions for additions or corrections to this manual should be directed to an ASC applications analyst or to:

Operations - User Manual
Alabama Supercomputer Center
686 Discovery Drive
Huntsville, AL 35806

This manual is supplemented by a set of policies, which cover various aspects of the Alabama Research and Education Network. AREN policies are available from ASC applications analysts and are available online using **asninfo** on the Cray SV1.

TABLE OF CONTENTS

Figures

The Alabama Supercomputer Center

The Alabama Supercomputer Center (ASC) provides state-of-the-art high performance computing capabilities to universities, colleges, and K-12 schools in the State of Alabama. Commercial accounts are also available. The Alabama Research and Education Network (AREN) is a statewide network that connects clients to the Internet and to ASC's computing resources via high-speed telecommunications links.

The Alabama Supercomputer Center (ASC) provides high performance computing resources to state academic users, state government agencies, national industrial users, and federal government agencies. Networking services are also provided, including access to ASC high performance computing resources, Internet access, World Wide Web services, and training. ASC resources, including a Cray SV1 supercomputer, are accessed through the Alabama Research and Education Network (AREN), a statewide high-speed network installed and maintained by ASC.

The Alabama Supercomputer Authority (ASA) operates ASC. ASA is a public state nonprofit corporation that develops, maintains, and operates the Alabama Supercomputer Center and the Alabama Research and Education Network. Technical services are provided through professional services and facilities management contractor Nichols Research Corporation (NRC).

The basic configuration of the Alabama Supercomputer Center is shown in Figure 1. The centerpiece of ASC is a Cray SV1 parallel vector processing supercomputer with a tightly coupled I/O subsystem. It has sixteen CPUs, 2048 million-words of central memory and 480 gigabytes of RAID-3 fibre channel disk storage. The Cray SV1 interfaces to the remainder of the network through a fiber distributed data interface (FDDI) ring in the Alabama Supercomputer Center (see Figure 1).

Other high performance computing resources at ASC, available through the network, include Sun, SGI, and Linux workstations. They are used as web, email, domain-name, and license servers.

The growing network comprises nodes connected through routers, leased DS1 (T1) (1.544Mbs), DS3 (T3) (45Mbs), and ISDN lines.

Except for periods of scheduled maintenance, the network is available 24 hours per day, every day of the year. The center is staffed 24 hours per day, every day except Christmas and New Year's Day.

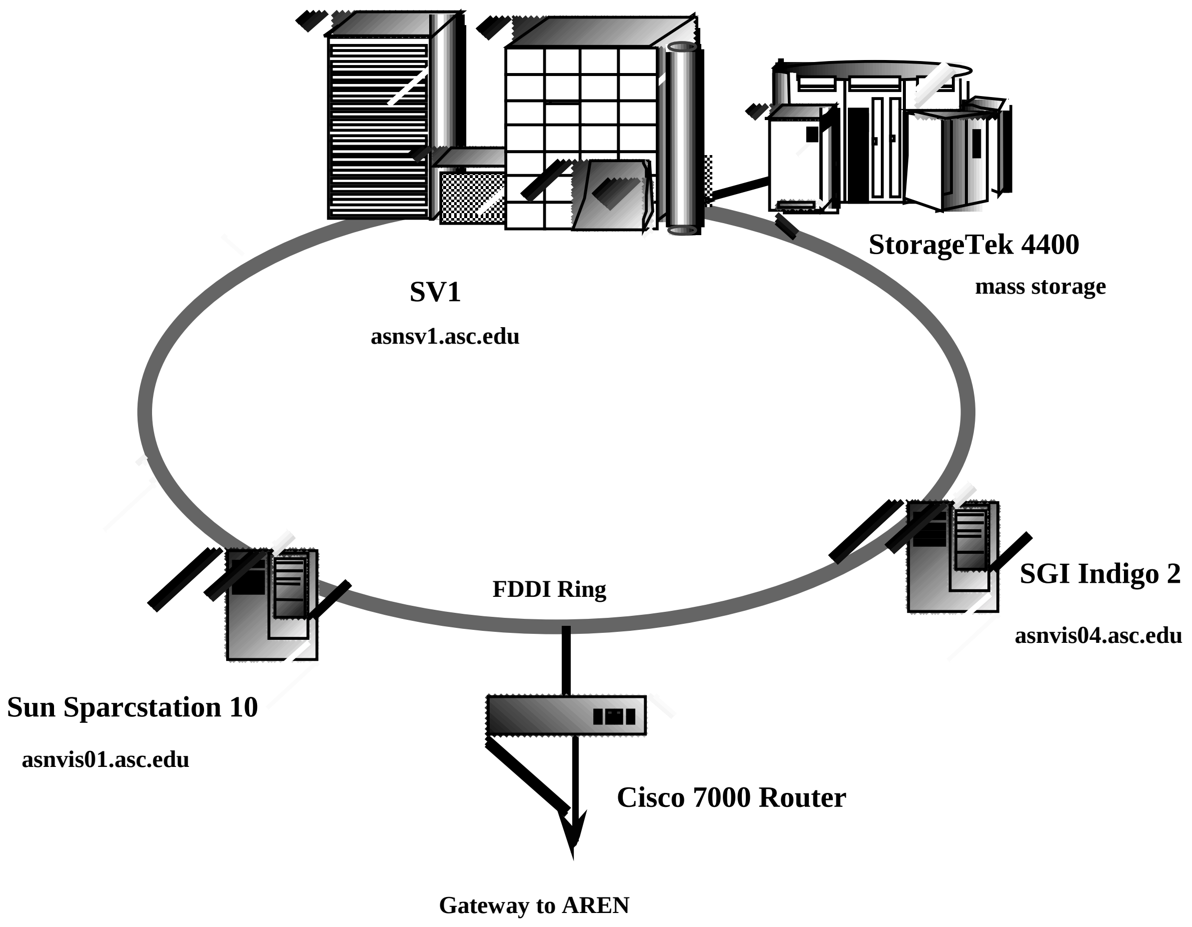

Figure 1. Alabama Supercomputer Center Primary Equipment

CRAY SV1 High Performance Vector Processor

The ASC Cray SV1 (see Figure 2) consists of sixteen processing units (CPUs), 16 gigabytes of memory, and 480 gigabytes of RAID-3 fibre channel disk storage.

Each Cray SV1 processor can produce results at a rate of ***1.2 Billion Floating Point Operations per second*** *(GigaFLOPS). A rate of 19.2 GigaFLOPS can be achieved by the use of all 16 CPUs in parallel.*

CPU

The sixteen CPUs are identical and can process in scalar or vector mode.

Clock period	4 nanoseconds
Vector units	32
Maximum result rate	1.2 GFLOPS

Groups of four SV1 processors can be configured as multi-streaming processors (MSPs). An MSP has a peak processing rate of 4.8 GigaFLOPS.

Memory

Word length	64 bits
Cycle time	4 FLOPS/clock
Address space	2048 million words
Max bandwidth/channel	6.4 GB / second

RAID-3

The RAID-3 fibre channel disk has a capacity of 480 gigabytes.

Vector Units

The Cray SV1 has 32 vector units per processor.

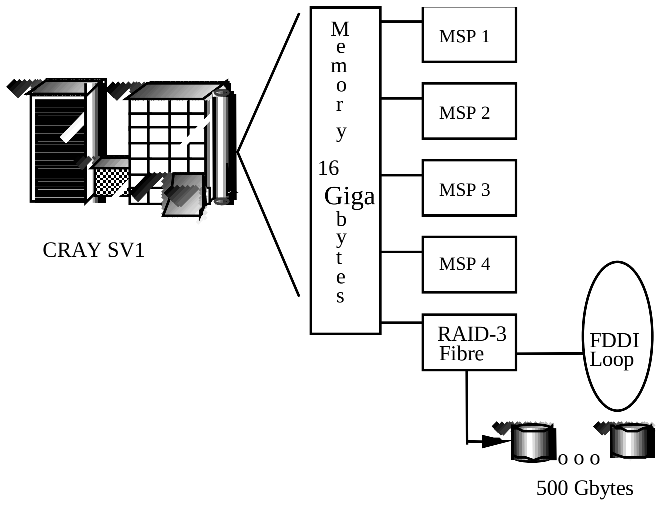

Figure 2. Cray SV1 Architecture

StorageTek 4400

With the advanced computational power available at Alabama Supercomputer Center, users need storage capabilities that provide adequate access speed and allow storage across multiple computing platforms. StorageTek's 4400 Automated Cartridge System (ACS) meets these challenges with an efficient implementation of systems managed storage. The ACS 4400 is a fully automated information storage system that mounts and unmounts 1/2-inch 18-track cartridges. The PowderHorn, a high performance freestanding robot, achieves quick and consistent data access. PowderHorn uses a balanced "H-arm" with dual "hand" mechanisms on a rotating turntable to achieve up to 350 cartridge exchanges per hour. Mounted on each hand is a solid-state camera, which scans the bar-coded cartridge volume label and mounts the cartridge in the appropriate drive.

ASC Cray SV1 users access the StorageTek (STK) through the use of Cray's Data Migration Facility (DMF). DMF ensures the availability of file system space by moving selected files from online disk to the STK ACS. This change in residence is transparent to users because the files remain cataloged in their original directories. When a file is migrated to the offline storage system, its i-node (information node/directory entry) remains online, but the data is moved offline. Files can be migrated either by explicit user request or by the automated space management components of the DMF. These components automatically monitor and adjust the contents of the file system based on criteria specified by the system administrator. The ACS is an offline storage facility and cannot be considered a file backup option. The deletion of a file results in the removal of the i-node and, therefore, the permanent removal of the migrated data.

The ASC Cray SV1 file system ***/dmf_stage*** is provided for user files and is under the automated space-management facility, DMF. When an ASC Cray SV1 new user account is created, a directory on the ***/dmf_stage*** file system is created for that user. Any user files placed in the user's ***/dmf_stage*** directory will be automatically migrated to tape by DMF. The ***/dmf_stage*** file system hierarchy consists of the file system ***/dmf_stage*** and subdirectories indicating a user's group or university association. For example, a user with username uabxxx01 would have the following DMF staging directory: ***/dmf_stage/uab/uabxxx01*** where uabxxx01 is the name of the user's DMF staging directory. The user can also explicitly migrate files from ***/dev/dsk/home*** to ***/dev/dsk/dmf_stage*** by issuing the command:

dmput [options] ***filelist***	-marks a file for migration
dmget ***filename***	-retrieves an offline file from storage

Migrated files are not charged against a user's disk quota. A user can immediately release disk space for usage by including the **'-r'** option to the ***dmput*** command. A user must have enough disk space remaining to retrieve a file from storage and still remain under the user's disk quota if that user is retrieving a file from a directory other than the user's ***/dmf_stage*** directory. Consult the man pages for ***dmput*** and ***dmget*** for examples and further explanation of these commands.

This offline status of a migrated file is indicated by a letter **'m'** in the leftmost column of a long format directory listing. The long format directory listing is obtained by issuing the '**ls -l**' command.

For example:

```
% ls -l
mrwxr-----   1 uabxxx01 uabgroup        3330 May 18  1999 datfile1.dat
mrwxr-----   1 uabxxx01 uabgroup        1863 May 19  1999 datfile2.dat
-rwxr-----  . 1 uabxxx01 uabgroup   1702976 Aug 19  1999 a.out
```

The **'m'** in the left column of the listing indicates that the first two files are migrated to offline storage while the third file, **a.out**, is not migrated and resides on hard disk space.

Available Software

Several software packages and libraries are available on ASC systems, including the IMSL libraries and application packages such as UNICHEM, ABAQUS, MSC/NASTRAN, GAUSSIAN 98, and others.

Design Analysis

ABAQUS
A general purpose finite element code for analysis of structures subjected to static or time-varying loads. Abaqus is formulated to handle non-linear structural problems easily.

EMAS
An electromagnetic simulation and analysis system for a wide range of DC, AC, and time domain applications, with comprehensive sensitivity to nonlinear and anisotropic materials.

LS-DYNA3D
Explicit finite element program used to analyze the nonlinear dynamic response of inelastic structures with fully automated contact capabilities. A post-processor for LS-DYNA3D, LS-TAURUS, is also available.

MSC/NASTRAN
Widely used finite element analysis program for structural mechanics. NASTRAN has a very broad range of capabilities.

SINDA/G
SINDA, the Systems Improved Numerical Differencing Analyzer, is a software system which has capabilities that make it suited for solving lumped parameter representations of physical problems governed by diffusion-type equations.

Biotech/Computational Chemistry

AMBER
Amber is a suite of programs for performing a variety of molecular mechanics-based simulations on machines ranging from workstations to supercomputers. Amber uses LEAP as its graphical interface.

AMSOL
Semi-empirical chemistry code designed to model molecules in solution.

CADPAC
The Cambridge Analytic Derivatives Package is an ab initio suite of programs which offers calculations of polarizabilities, dipole moment derivatives, and forces constants at the MP2 level using fully analytic approaches.

DeFT
DeFT is a program for computations on molecules using DFT theory.

DGAUSS
DGauss is a high-accuracy, high-performance computational chemistry package that uses density functional theory to predict molecular structures, properties, and energetics.

GAMESS
The General Atomic and Molecular Electronic Structure System is an ab initio quantum chemistry code.

GAUSSIAN 98
The pre-eminent ab initio quantum mechanics package. Enhancements relative to Gaussian 94 include advanced optimization algorithms, a fast multipole method and sparse matrix techniques for linearizing computational cost, and implementation of the ONIOM layering method.

GROMOS 96
A molecular mechanics and dynamics program for simulating molecules, liquids and biomolecules.

JAGUAR
High performance quantum mechanics package with GUI. Jaguar scales more favorably to large numbers of atoms than many other ab initio codes.

MOPAC
MOPAC is a general purpose, semi-empirical molecular orbital program for the study of chemical reactions involving molecules, ions, and linear polymers.

MNDO97
MNDO97 is a semi-empirical program for studying the electronic and structural properties of molecules.

POLYRATE
POLYRATE computes chemical reaction rates using variational or conventional transition state theory. POLYRATE is designed to compute additional information using data generated by other programs as its input. (e.g., the results computed by GAMESS or Gaussian can be used as the input to POLYRATE).

UNICHEM
Graphical chemistry package for pre- and post-processing of GAUSSIAN, CADPAC, DGAUSS, and MNDO calculations.

Crystallography

CNS
Crystallography & NMR structure determination package.

CRYSTAL 98
A program for ab initio and density functional theory calculations on crystals.

Computational Fluid Dynamics

FIDAP
FIDAP is a computational fluid dynamics (CFD) package that analyzes fluid flow including heat and mass transfer. It includes comprehensive pre- and post-processing capabilities and a flexible and powerful solver.

FLUENT
FLUENT is a general-purpose finite volume code useful for modeling complex flows ranging from incompressible to highly compressible (up to Mach 10). Included models can accurately predict laminar, transitional and turbulent flows, various modes of heat transfer, chemical reaction, and multiphase flows. Both structured and unstructured meshes are supported.

GAMBIT
GAMBIT is available on local workstations as a pre-processor/grid builder for FLUENT and FIDAP. It supports structured and unstructured meshes and can import ACIS and IGES/CAD geometries. Some low level boundary conditions can also be applied within GAMBIT.

GASP

GASP is a structured, multi-block CFD flow solver which is applicable to compressible flow fields approximately Mach 0.2 and greater and includes finite-rate chemistry cases.

GRAPE3D

GRAPE 3D generates 3-dimensional computational grids about arbitrary configurations. GRAPE 3D employs a fast iterative Poisson equation solver to generate 3-D grids about arbitrary shapes.

INS3D

INS3D solves the incompressible Naiver-Stokes equations in 3-dimensional generalized coordinates for both steady state and time varying flow. Users must obtain their own copy of INS3D from NASA.

MM5

MM5 version 2 is used for weather prediction and climate modeling. This version includes three additional cumulus parameterizations, a new planetary boundary layer scheme, a radiation scheme, and a multi-layer soil temperature predication scheme.

General Math and Statistics

IMSL
The IMSL Libraries are a comprehensive resource of more than 900 FORTRAN mathematical and statistical subroutines for scientists, engineers, and mathematicians.

PDE2D
PDE2D solves quite general nonlinear, time-dependent, steady-state, and eigenvalue systems of partial differential equations.

Software packages are added to the system from time to time. For a current list, contact a local site analyst or the ASC Help Desk.

SLATEC
SLATEC is a math library that supports such calculations as: arithmetic, error analysis; elementary and special functions; elementary vector operations; solutions of systems of linear equations; and much more.

Compilers/Programming Aids

CAL- Cray Assembler
Standard Cray Assembler

F90 - Cray Fortran 90 compiler
Cray Fortran 90 autotasking compiling system.

FMP- Autotasking Preprocessor
Standard Cray autotasking preprocessor.

MPI
Message passing library.

SCC - Cray C Compiler
Standard Cray C Compiler

XProf
X Windows based code profiling tool.

Visualization

GNUPLOT
Gnuplot is a command driven interactive function plotting program. Gnuplot plots any number of functions, built up of C operators, C library functions, and intrinsic operations such as exponentiation.

LS-TAURUS
LS-TAURUS handles LS-DYNA3D output and displays it in multiple graphic forms for the user to better evaluate the results of the LS-DYNA3D analysis.

NCAR Graphics
A set of FORTRAN library routines for generating graphics from a user application.

XIMAGE
2-Dimensional image display and manipulation program for X-Windows.

Operating Systems

UNICOS

Sun OS

IRIX

AIX

Technical Support for Users

An automated problem management system is operated at the Alabama Supercomputer Center to provide tracking and accountability for problem resolution.

There are five basic elements to the ASC support program:

- *On-Site Applications Analysts*
- *Help Desk Service*
  - ***1-800-338-8320***
  - ***Email: helpdesk@asc.edu***
- *Problem Management System*
- *Documentation*
- *Training*

Support for ASC users combines central site (ASC) support with campus based applications analysts and is supplemented with training programs. This support allows the user to process productively as rapidly as possible. Ongoing support to overcome problem areas and in mapping high performance computing technology into the researcher's specific area of study is provided.

Applications analysts based at selected sites are available to provide training and support to the local user community.

Specialized local training is available to enhance user productivity.

A manned help desk is available 24 hours a day to assist with problem solving and to answer user questions about the status of the ASC systems and AREN. The Help Desk is accessible through a toll-free number, 1-800-338-8320, as well as through electronic mail (helpdesk@asc.edu) on the network.

An automated problem management system is operated at the central site to provide tracking and accountability for problem resolution.

A full set of documentation for ASC resources is available at selected nodes and the central site. Users may also purchase vendor manuals if desired. Electronic versions of ASC resource documentation is available online.

Three kinds of technical support are provided by the Alabama Supercomputer Center for AREN users: local campus applications analysts, toll-free "hotline", and consultation support from ASC central site analysts

Local campus applications analysts provide leadership in the growth of high performance computing technology at their site in their area of responsibility. They can assist the universities in the development of courses, collaborate in research efforts, and coordinate local support requirements.

The focal point for technical support is the local applications analyst. These analysts work as a team to provide the following services to both educational and industrial users across the state:

General Support: Analysts provide assistance in establishing user accounts, job control language, program compilation and execution, and other user support as needed.

User Training: Analysts provide introductory lectures, classroom training, and one-on-one instruction.

Application Program Support: Analysts provide support for conversion of programs, optimization and vectorization of code, use of application packages, and resource management.

Outreach Support: Analysts assist in promotion of ASC resources to potential academic and industrial users, through formal technical presentations, demonstrations, benchmarking of codes, and technical consultation.

Contact Information: If you do not know how to reach the analyst assigned to your campus or institution, please call the Help Desk at (800-338-8320.

Online Help

Various online help facilities are available. UNICOS or UNIX information can be obtained with the **man** command:

man *title*

or

man -k <*keyword*>

The **man** command locates and prints the entry named *title*. The title is entered in lowercase. The following example reproduces the description of **man** on the standard output:

man man

To get information on the f90 command, type:

man f90

Using Help Desk Electronic Mail Facilities

Procedures have been established to MAIL a user's problem to the Help Desk directly from the prompt.

To send a problem or a question to the Helpdesk, enter **helpdesk** and follow the instructions.

Sample request for help on the Cray:

```
asnsv1$ helpdesk

    Welcome to the ASC HelpDesk Facility


Please enter your question or problem.
To end input and send this text as mail, enter a
 <Control>D
Your message must contain at least one Carriage
Return.

    ....Description of help needed goes here...........

  <cntl>D

Please wait to determine active helpdesk host
Mail delivered to helpdesk@asnmail.asc.edu.

    Thank you for using the HelpDesk Facility.

asnsv1$
```

Requesting an ASC Account

*A**cademic Block:** Available computing resources are allocated to the universities in accordance with policies set by the Board of Directors of the Alabama Supercomputer Authority. The allocation for each university is established semi-annually by the CEO of the Alabama Supercomputer Authority.*

***Commercial Block:** Available computing resources are allocated for commercial and non-Alabama academic users by the Alabama Supercomputer Authority.*

The Alabama Supercomputer Authority has designated two categories of accounts:

Academic Accounts are requested by the individual universities according to university developed policy. The local applications analyst can explain the procedure for obtaining an academic account at a particular university.

Commercial Accounts include commercial and non-Alabama university research activities. All commercial accounts are coordinated through the Huntsville ASC office. The ***Agreement for Computing Services*** form must be completed. The local applications analyst can provide assistance in coordinating this agreement.

Charges are not assessed for the use of the Cray for unsponsored research by Alabama's public educational institutions. However, sponsored use by universities is nominally charged at 80% of the prevailing commercial rate. **Some software packages, such as MSC/NASTRAN, have a usage royalty charge which may apply to some accounts**.

Each user must have a separate account on the Cray. To request an account, the user should submit an ***AREN Account Activity Request Form*** to the local applications analyst. These forms are available from the analyst or the campus designated account authority, an individual who administers the institution's allocation of computing time. The completed form is forwarded to the ASC account administrator for processing. Once the account is established, the user is notified by the applications analyst or the account authority.

ASC Accounting System and Procedures

The computer accounting system for the Alabama Cray SV1 supercomputer uses standard UNICOS accounting data with several enhancements. The enhancements provide the following functions:

•Dollar limits on accounts
•Current and total resource usage display
•Proprietary program surcharge handling
•Month-end billing information

An account will automatically be prohibited from further processing when the account limit is exceeded. When this occurs, a user will not be allowed to begin an interactive session or batch job and will receive a message stating that the dollar limit has been exceeded for the session.

A user applying for an account will be asked to specify which surcharged programs will be used, and authorization for those programs will be added to the account. When an account uses one of the surcharged programs, the accounting system looks up the associated surcharge rate in an internal table and adds it to the dollar amount charged to the account.

The Alabama Supercomputer Authority reviews month-end accounting report files. Actual dollar amounts from these reports are used by the Authority's accounting system to compute the amount billed to commercial users and users with software royalty charges. Unsponsored academic research accounts are not billed directly but are limited by the account dollar limit mechanism to ensure an equitable distribution of the high performance computing resources.

The accounting system and related procedures handle all the administrative requirements for accounting. Procedures in place for adding new accounts, deleting unused or expired accounts, and modifying existing accounts include actions to handle such situations as increasing the dollar limit, adjusting for charge backs, changing rate schedules and priority factors, changing expiration dates, and handling security in case of forgotten passwords. The applications analyst or account authority for an account can help solve problems relating to such issues.

Tracking Resource Consumption

A user may review the charges accrued to an account at any time by issuing the **usage** command. The account limit, previous dollar usage, and the account month-to-date totals are displayed.

Resources are assigned and controlled by the ASC accounting system which tracks and logs usage of various system resources. Usage totals are updated as follows:

Every account is given a dollar amount limit for Cray usage. It is the user's responsibility to control resources and to monitor the amount of the allocation which has been used.

Resource	**Frequency**
CPU usage	end of job
Memory Residency	end of job
I / O Time	end of job
Connect Time daily	
Pages/Plots	daily
Total Dollars Used	end of job & daily
Previous Months $ Used	monthly

The charge for service is based on a "recorded hour" which is calculated by the accounting system from CPU hours, average memory residency, and input/output time (disk accesses and data I/O). The formula used is:

CPU Hour
+1/10 memory Mwords*Hour
+1/100 I/O Hours
+1/10 Connect Hours
=Recorded Hour

Service Charges

Service charges for the Cray SV1, measured in Recorded Hours, are based on CPU time, interactive connect time, memory usage, and disk access. The monetary charge per Recorded Hour is stated in the AREN Policy, ***ARENP04: Processing Charges.*** NQS queue structure and limits are shown in Figure 3. Interactive limits are shown in Figure 4. The command **nqslimits** outputs the current queue structure and limits.

Express processing is charged at 1.4 times the base rate. Interactive and prime processing are charged at the base rate, and off-hours processing is charged at 0.6 times the base rate.

Queue	Pri	Run	Nice	CPU/req	Mem/req	File/req	File/proc
small	50	16	2	144000sec	16mw	1gb	1gb
medium	40	16	4	324000sec	32mw	4gb	4gb
large	30	8	6	324000sec	64mw	8gb	8gb
huge	20	2	8	324000sec	256mw	16gb	16gb
express	60	4	0	3600sec	32mw	1gb	1gb
sysadm	10	4	2	144000sec	32mw	4gb	4gb

Figure 3. NQS Queue Structure and Limits

	Pri	Nice	CPU/req	Mem/req	File/req	File/proc
INTERACTIVE	0	0	600sec	16mw	128mb	128mb

Figure 4. Interactive Limits

The *usage* Command

*Accumulated resource usage on the Cray can be displayed with the **usage** command. Options allow users to show actual CPU utilization or account dollar balance.*

The ***usage*** command presents session cost information to the user. It can provide resource usage detail or current month-to-date and account limit information. It is available during each interactive session and provides account information for the session.

Options:

usage [-c] [-s]

-c	show actual CPU utilization
-s	show recorded hour utilization

Effective Resource Usage

There are several important steps a user can take to reduce computer usage costs and to use the supercomputer efficiently. These include adjusting user priority and type of access, monitoring offline resources, and using good programming techniques.

Users are urged to use the least expensive priority that is effective for their requirements. In addition, careful consideration should be given to whether a job is run in interactive or batch mode.

Offline editing of programs (on a local machine or a personal computer) considerably reduces connect time on the Cray and is highly recommended.

Online storage should be periodically checked for obsolete or seldom used files. Cray home directories are limited to 50 MB. Printing can be reduced by reviewing the output file using the terminal screen and by discarding runs that are incomplete or, for some other reason, do not need to be printed.

If possible, users should run small test jobs before running larger production jobs; this will minimize debugging costs. The usual care should be given to ensure that input data is accurate and that careful programming techniques are used.

Alabama Research and Education Network Overview

Access to the Cray SV1 is available through the Alabama Research and Education Network. The Alabama Supercomputer Center is connected to the AREN wide area network which extends throughout the state.

The Alabama Research and Education Network not only connects each node in the network directly to the SV1 but also interconnects all the nodes to allow resource-sharing and communications among them. To accomplish this, the supercomputer is connected to a 100 Mb/sec FDDI ring, which in turn is connected to a 10 Mb/sec local area Ethernet. This Ethernet is connected to the AREN wide area network which extends throughout the state through the use of leased DS1, DS3, and ISDN lines.

The network uses the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite. TCP/IP provides interactive access, file transfer, and electronic mail service not only to the SV1, but also among all nodes on the network. It also provides a gateway, which ties AREN into the global Internet network.

Users should contact their applications analyst if assistance is needed to dial into the network.

Many access paths are available to ASC. Access from the campuses is usually through the campus network; any user with access to an Internet host can access ASC and AREN. Most campus networks have dial-up lines available. The dial-up line numbers are available from the campus applications analyst or campus network administrator.

General Network Capabilities

AREN users can communicate directly with the Cray, as well as any other machine on the network which supports TCP/IP.

Users have the following capabilities over the network:

telnet

The **telnet** command allows users on any computer connected to the Alabama Research and Education Network to interact with computers at the Alabama Supercomputer Center as if they were directly connected to them. This is the primary method for interactive access to the Cray. For security reasons, only nodes with Cray accounts have telnet access to the supercomputer.

ftp

The **ftp** command allows users to transfer files between machines.

mail

The **mail** command allows users on any machine connected to the AREN to send electronic mail to any other user on the Internet that has been issued an account/e-mail address.

lpr

The **lpr** command allows users logged into a UNIX based machine such as the Cray or the Sun to print files at a remote site. Contact the campus applications analyst to add remote printers or to get a list of available remote printers.

WWW Access

ASC provides users with access to the World Wide Web (WWW**.** ASC's web server is located at url: **http://www.asc.edu**

Connecting to the ASC Computing Resources

*In order to connect to a machine on the AREN the user **MUST** have a **user_id/account and password** on that machine.*

When a user dials up to a server, the user must enter their user_id and the appropriate password. See the campus application analyst if any problems occur. After users are logged in or connected to the server, a **local>** prompt is displayed**.** To connect to a host, enter:

local> **telnet hostname**

where *hostname* is the name of the login system.

Example:

local>**telnet asnsv1.asc.edu**

Access from the Nodes

Any processor connected to the AREN or to a connected network can access the ASC facilities if it supports the TCP/IP protocol.

Note: The IBM 3270 systems at TSU do not allow use of the **telnet** command unless the user is using or emulating an IBM 3270 terminal.

When using commands involving a host name, such as **telnet** or **ftp**, users should always specify the full name, e.g. **asnsv1.asc.edu.** This should also be done when connecting from a terminal server.

Communication in Dial-up Mode

Dial-up to the host at AREN nodes requires a host account. Dial-up to the terminal servers allows the user to connect to any system recognized by AREN. The AREN servers allow connection to any ASC system and most of the connected campus networks. The AREN servers require the user to enter an identifier and a password before connecting. See the campus Applications Analyst for this information.

Most sites support up to 33,600 baud duplex modem connections, 8 bit, 1 stop bit, and none/no parity.

WWW Access

The Alabama Supercomputer Center maintains its own World Wide Web site.

The URL for the ASC web server is:

http://www.asc.edu

The ASC homepage provides links to related sites on the web. ASC also hosts web servers for other activities, such as the Alabama Virtual Library.

Connecting Hardware to the Network

Permission must be received ***before*** *any hardware can be directly connected to the AREN.*

Universities are permitted to connect certain mainframes and other networks to the AREN.

There are several guidelines to keep in mind:

- Network connections must be made through a router.
- Internet names and addresses must be coordinated with the Alabama Supercomputer Center.
- Please coordinate the connection request with the campus applications analyst or with the ASC service manager.

ASC Cray SV1 Operating System

The ASC Cray SV1 runs under the UNICOS operating system, which is the Cray Research, Inc. version of UNIX.

The interactive UNIX operating system was developed by AT&T and has been extended and enhanced at the University of California at Berkeley. UNICOS is based on the AT&T UNIX System V with some enhancements from Berkeley UNIX (bsd 4.3). As such, it is very much like UNIX operating systems on a wide variety of other computers.

UNICOS provides the standard UNIX commands, libraries, and features, such as user shells, pipes, tees, and filters. Also included are text editors (ed, ex, and vi), communications programs (telnet, ftp, and mail), compilers (C, C++, and FORTRAN90), and the Network Queuing System (NQS) - a Cray enhancement of UNIX to provide "batch" operation. NQS allows users to create a file of commands for the computer to execute at a later time rather than simply typing in the commands one by one from a terminal. Effective use of the Cray SV1, particularly for large jobs, requires use of the NQS system.

Complete UNICOS documentation is available in the manuals listed in the *Documentation* section of this manual. Many popular UNIX System V guides and textbooks also provide valuable information and are generally applicable to UNICOS.

Most documentation for UNICOS and for the additional programs provided at the Alabama Supercomputer Center can be accessed online via the **man** command. See the *Obtaining Assistance* section of this manual for information about getting help.

Using UNICOS interactively is described in the following sections. The NQS batch system is designed to accept the same commands as for interactive use. You can prepare a file for submittal to the NQS system with one of the Cray text editors and then place it into an appropriate batch queue.

Dataset Naming Conventions in UNICOS

The same rules apply to both file and directory names in UNICOS, which is based on UNIX System V.

- Nearly any keyboard character may be used in file and directory names. It is best to avoid the characters that UNICOS uses in other situations. such as **/ \ " ' * ; - ? [ ] ( ) ~ ! $ { } <** and **>** which all potentially can create confusion.
- Names may be from 1 to 254 characters long. Periods and underlines may be used to substitute for blanks (which are not allowed) to clarify what the names mean. For example, a documentation file might be designated **read.me** or **read_me**.

UNICOS is CASE SENSITIVE! *Uppercase is distinguished from lowercase:* ***prog.for*** *is not the same file as* ***Prog.for****, for example.*

Directories in a UNIX system are organized in a tree structure. The root directory is designated /. Users on the system have their directories and files placed in a branch of the root corresponding to their node. Other important directories, such as ***sys*** and ***bin***, also branch directly from the root. These directories are designated **/home**, **/bin**, and **/sys**, and are illustrated in Figure 6.

Subdirectories are indicated with a / preceeded by the name of their parent directory. If there is a user subdirectory called **asn** in **home**, for example, the full designation for that subdirectory, starting from the root, would be **/home/asn**. The user **asndsc01** might have organized FORTRAN programs into a subdirectory called **fort**, and one of those programs might reside in the file **prog.f** within the **fort** subdirectory. The full path designation for the file **prog.f** thus would be:

/home/asn/asndsc01/fort/prog.f

Any given directory has a parent directory, in which it is a subdirectory. Shorthand for the parent directory is **".."**.

To get to the directory **mail** from **fort** in Figure 7, one uses the "***change directory***" command **cd** in UNICOS:

```
cd ../mail
```

The shorthand designation, **"."**, represents the current directory.

To copy all files with extension **.f** from the parent directory to the current directory, use the following command:

```
cp  ../*.f   .
```

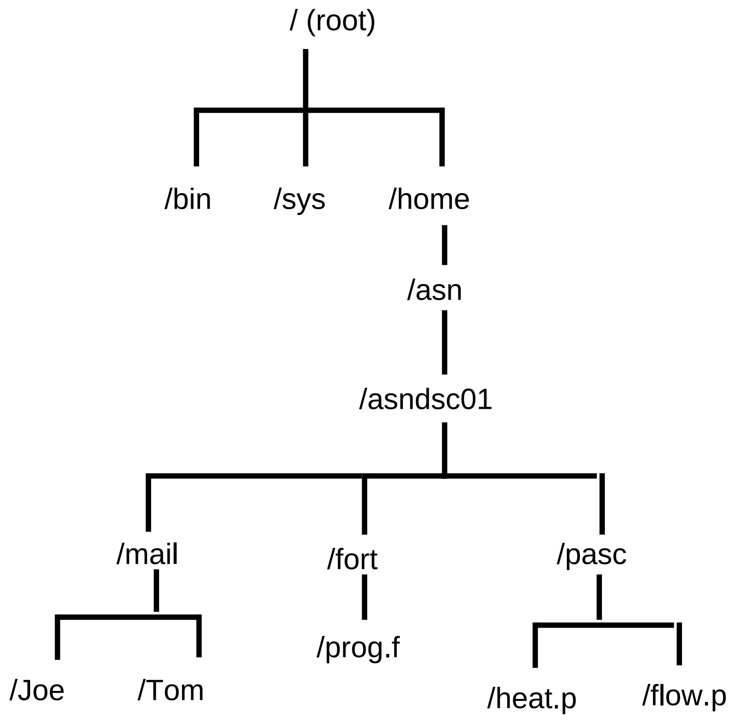

Figure 5. Sample UNIX Tree Structure

ASC UNICOS File Organization

The files on the ASC SV1 are organized as diagrammed in Figure 7. Users may locate files on **/home**, **/dmf_stage**, and the **/tmp** file systems. Files on **/home** and **/dmf_stage** are moved between online and offline storage by Cray's Data Migration Facility (DMF) and the Storage Tek tape system. Each user has a symbolic link in their **$HOME** directory to their own directory on **/dmf_stage**.

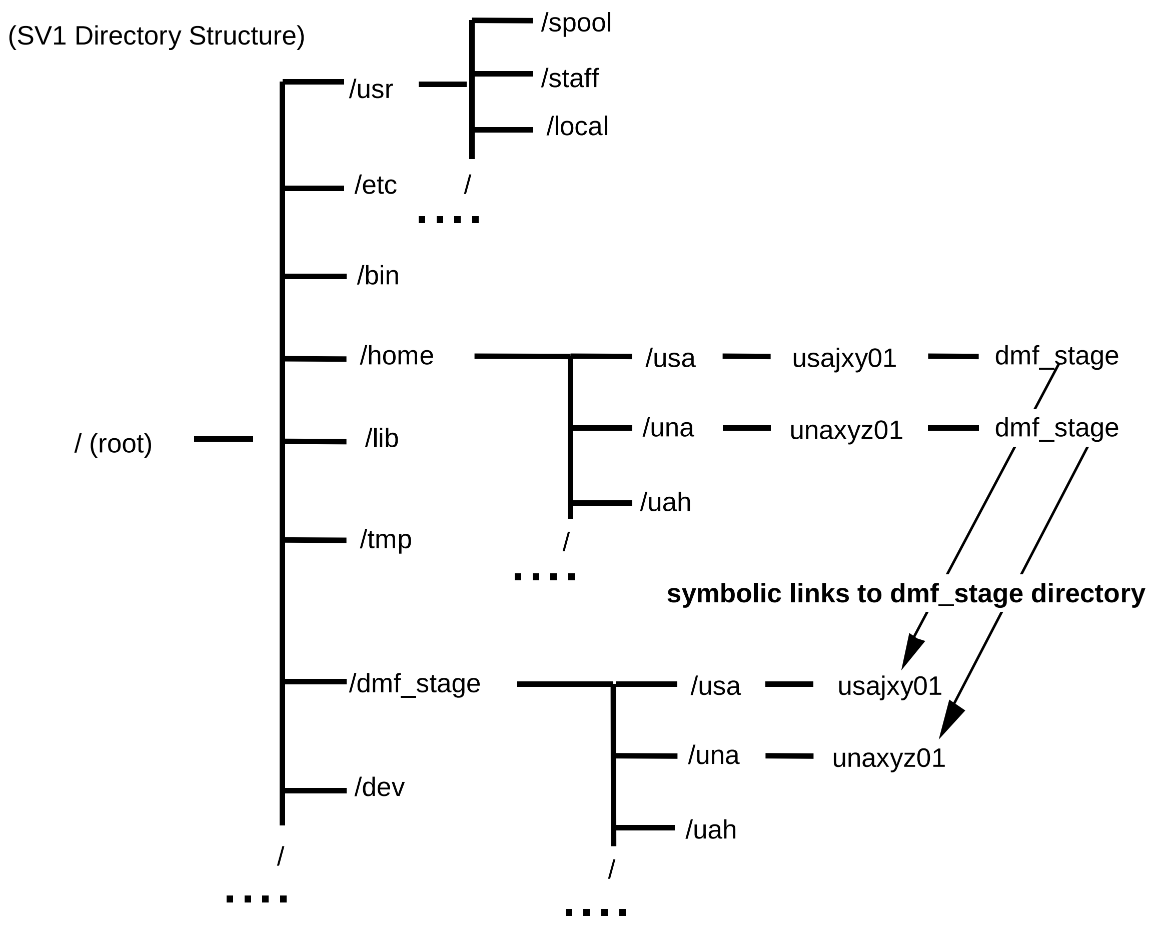

Figure 6. UNICOS File Organization

Manipulating Files and Directories

UNIX/UNICOS commands are used to create, move, copy, or delete files and directories. Each command must be in lower case as shown.

cd

cd without arguments puts the user in the user's home directory. With a directory name as an argument, the command moves the user to that directory.

cp

cp makes copies of files in two ways.

cp filea fileb

makes a new copy of **filea** and names it **fileb**.

cp [list of files] directory

puts copies of all the files named into the directory named. Contrast this to the **mv** command which moves or renames a file.

ln

ln creates a link between files.
Example:

The following links the existing file example.c to ex.c.

ln example.c ex.c

The following creates symbolic links.

ln -s /usr/include incl

See the online **man** pages for many other ways to use **ln**.

mkdir

mkdir makes a new subdirectory in the current directory. Example:

mkdir fort

makes a subdirectory called **fort**.

mv

mv moves or changes the name of a file. Examples:

mv filea fileb

changes the name of **filea** to **fileb.** If the second argument is a directory, the file is moved to that directory.

One can also specify that the file have a new name in the new directory:

mv filea direc/fileb

would move **filea** to directory **direc** and give it the name **fileb** in that directory.

Contrast this to the **cp** command which copies files.

pwd

pwd returns the name of the current working directory. It simply tells you the current directory.

rm

rm removes each file in a list from a directory. Option **-i** causes **rm** to inquire whether each file should be removed or not. Option **-r** causes rm to delete a directory along with any files or directories in it.

rmdir

rmdir removes an empty directory from the current directory. Example:

rmdir fort

removes the subdirectory named **fort** (if it is empty of files).

To remove a directory and all files in that directory, either remove the files first and then remove the directory or use the **rm -r** option described above.

Logging in to the Cray SV1

Interactive access to the CRAY SV1 is obtained through the **telnet** command.

Example of a Cray SV1 login:

```
% telnet asnsv1.asc.edu
Trying...
Connected to asnsv1.asc.edu.
Escape character is '^]'.



Cray UNICOS (sn9993) (ttyp017)

UNICOS 10.0.0.6 is now running with SV1 rev. B CPU's. Users will
need to recompile to take advantage of the increased performance.

login: asntrd01
Password:

Active label set to : level0,none

Last successful login was : Thu Feb  3 13:24:43 from gcwbackup.asc.edu

      ****************************************************************
      *      Welcome to the Alabama Supercomputer Network            *
      *          >>>>>   UNICOS 10.0.0.6   <<<<<                     *
      ****************************************************************

      This is a State of Alabama computer facility managed by
      the Alabama Supercomputer Authority and its contractor
      Nichols Research Corporation.  The system is for use by
      State of Alabama educational institutions and authorized
      commercial accounts.

      In accessing the system, users consent to the monitoring
      of usage for purposes of accounting and the detection of
      unauthorized access.  Please note that access to this facility
      must be specifically authorized by ASA and that unless you are
      so authorized, access and any other use may expose you to
      criminal and/or civil proceedings.

*****************************************************************************
```

```
        The batch queue limits have been modified since the upgrade to
        UNICOS 10.0.0.6. The total batch job limit is four (6) jobs per user.
        Individual queue limits (per queue, per user) are as follows:

                Queue               Limit
                -----               -----

                small                 6

                medium                6

                large                 4

                huge                  2

                express               4

                sysadm                4

***********************************************************************
You have mail.

asnsv1$
```

Logging Off of the SV1

To logoff the SV1, enter **exit** and a return.

```
asnsv1$ exit
Connection closed.
%
```

Login Profiles and Shells

*At login, the user's default shell is started, and the shell script is executed. The default shell for accounts is the Bourne shell, and the shell script is .**profile** in the user's root directory.*

The login shell can be changed to the C shell with the **csh** command. This will also change the login shell script to **.login**. When an account is created, the following scripts are copied into the user's directory from the default scripts in ***/usr/skel.*** The user may alter his login script as appropriate.

The Korn shell is also available on the UNICOS system. See **man ksh** for details.

/usr/skel/.profile	login script for Bourne shell, default
/usr/skel/.login	login script for C shell
/usr/skel/.logout	logout script for C shell
/usr/skel/.cshrc	additional script for C shell

Terminal Control

UNICOS supports many different types of interactive terminals connected through **telnet**. The default login script prompts the user for a terminal type. Valid entries include **vt100, vt220,** and many others.

Issuing the following command will list, screen by screen, the valid terminal names which may be used:

$ ls -C /usr/lib/terminfo/* | more

The UNICOS terminal control characters can be listed by entering **stty -a**. The defaults are:

interrupt = <CTRL-C>

quit = <CTRL-\>

end-of-file = <CTRL-D>

When using telnet from an IBM 3270 device, use the percent sign in place of the control key.

Frequently Used UNICOS Commands

Since UNICOS is, for the most part, a standard UNIX System V operating system, the usual set of commands is in effect.

cat

cat concatenates and prints the files given as arguments. The output goes to the standard output, which is usually the screen.
cat filea would print **filea** on the screen.
If no file is given, input is taken from the keyboard. A <CTRL-D> terminates keyboard input.
Often output is redirected with the operator >. Example:

```
cat filea fileb > filec
```

concatenates **filea** and **fileb** and places the result in **filec**.

chmod

chmod changes the file permission status of a file. Permissions may be granted to read, write, or execute the file, and the permission may be given to the user, the user's group, or to the world. When one uses the **ls -l** command, these permissions are listed at the left as a series of r's, w's, or x's, with - indicating that permission is not granted.

For example, -rwxrwxrwx indicates read, write, and execute permission is granted to all three groups; -r--r--r-- grants only read permission to each group. **chmod** changes the status of these permissions. The form is the following:

chmod [ugo] [+ -] [rwx] ***files***

u, g, or **o** stand for user, group, or others. The **+** or **-** indicate whether the permission is to be given or denied. And the **r, w** or **x** indicate whether read, write, or execute permission is to be given.

*Using **chmod** allows the user to change the permissions on directories since access to a file requires that:*

- *the user has access to the directory containing the desired file; and*
- *the user has appropriate access to the file itself.*

Example:

```
chmod ug+x filea
```

will give execute permission to the user and the user's group with respect to **filea.** When a new file or directory is created, the default permissions granted are those specified by the user's **umask** (or default permission mask). Each new ASC user has a umask defined in his default login script (.profile for Bourne shell or .login for the C shell). The user can change the **umask** value to suit the user's needs. See the **man** page on **umask** for details (use **man umask**).

Example: to give group read access to a file named **myprog.f** in a subdirectory called **fort** (i.e. give group read permission to the file **/uab/uabxyz01/fort/myprog.f**) use the following commands:

```
chmod g+x fort
```

sets execute permission for directory.

```
chmod g+r fort/myprog.f
```

sets read permission for file.

date

date outputs the date and time.

echo

echo repeats whatever text is given to it on the standard output. For example,

```
echo What's up, doc?
```

will print **What's up, doc?** on the screen (default for standard output). **echo** is useful for inserting messages in shell scripts, described later.

ls

ls lists the files in the current directory or the directory named as an argument. There are many options:

ls -a [directory]
lists all files, including files whose names start with a period.

ls -c [directory]
lists files by date of creation.

ls -l [directory]
lists files in long form: links, owner, size, date and time of last change.

ls -p [directory]
subdirectories are indicated by /.

ls -r [directory]
reverses the listing order.

l**s -s [directory]**
gives the sizes of files in blocks.

ls -C [directory]
lists files in columns using full screen width.

ls -R [directory]
recursively lists files in current directory and all subdirectories.

mail

mail, and mailx invoke an electronic mail system. **mail** and **mailx** can be used to both send and receive mail. Example:

mailx jim@asnuab.asc.edu

sends mail to jim at the node asnuab.asc.edu. The user names are arguments to the command. Users are prompted for a subject after the command.

The letter should be entered line by line and finished with a line containing only a period. Alternatively, the redirection operator < may be used to route a letter prepared with an editor to mail.

Example:

```
mail  jim  tom  <  letter
```

would send the previously prepared letter to users jim and tom.

man

man provides online documentation for all UNIX commands and utilities, making quite detailed descriptions instantly available. The commands or utilities are arguments to man. Example:

```
man cat
```

will give a summary of the use of the concatenate command.

```
man -k keyword
```

will give information on all commands relevant to the given keyword.

passwd

passwd allows the user to change his password. Prompts are issued in order to supply the old and new passwords.

paste

Merges same lines of files or subsequent lines of a file.

lpr

lpr is used to print files from the Cray to printers at the Alabama Supercomputer Center and certain printers at some of the universities. The basic form is:

lpr -Pprinter filename

See the section on printing for a complete description of this command.

sed

sed copies files (standard input by default) to standard output, edited according to a script of commands. The script is obtained from either the script operand string or a combination of the arguments from the **-e** script and **-f** sfile options.

sort

sort ***filename*** sorts the lines in the file in an extended alphabetical order, that is, the alphabet is extended to include special symbols and digits. The options for **sort** are the following:

sort -b [file]
ignores initial blanks.

sort -d [file]
uses a dictionary order, without special digits or symbols.

sort -f [file]
treats upper and lower case as equals.

sort -n [file]
sorts numerically.

sort -o filename [file]
puts the output in file **filename**.

sort -r [file]
reverses the sort order.

tail

tail prints the last part of a file given to it as an argument. The options are listed below:

tail +/-number b c [file]

tail starts printing **number** lines (or blocks or characters, with **b** or **c** added to the string) from the top (+) or bottom (-) of the file. Default is 10.

wc

wc (**w**ord **c**ount) counts lines, words, and characters in files. **wc file** will return counts for all three. The parameters **-l**, **-w**, and **-c** will limit the report to lines, words, and characters, respectively.

Example:

```
$wc filea
$20 100 500
```

```
20 lines, 100 words, 500 characters
```

who

who lists the login names of all users currently on the system, their terminals, and when they logged on.

who am I gives the same information about the user only.

write

write allows communication with other users currently on the system. The user names are arguments to the command. Each message should be terminated with **-o-** ("over") or **-oo-** ("over-and-out") and a CTRL-D to return to normal screen mode.

mesg y or **mesg n** may be set by any user to permit or not permit, respectively, **write** communications to come through.

Regular Expression Pattern Matching

grep

*The **grep** command matches patterns of ASCII characters in files fed to it as parameters. Positions of characters and multiple occurrences of a character may be matched as well as special characters.*

The form of the command is

grep *regular-expression filea fileb filec etc.*

The regular expression should be enclosed in single quotes if it contains one or more blanks. A regular expression contains the usual ASCII characters, but some characters have special meanings, depending on their location within the expression. The characters with special meaning are (. *** [ ] \\ $ and** ^). The period substitutes for any character in one position in the expression.

Example:

.abc will match with aabc, babc, 7abc, and so on, and **a.bc** matches a1bc, a2bc, azbc, etc.

Multiple occurrences of a character may be matched with an asterisk.

Example:

a*bc will match patterns with zero or more occurrences of **a**

bc abc aaaaaaaaaaaaaabc etc.

A backslash before a special character will remove the special meaning from the character, so that it can be matched for itself.

Example:

\* within a regular-expression will match * in that position.

Pattern Scanning and Processing Language

awk

The syntax for the ***awk*** utility is:

awk [-F ERE] [-v assignment] *program* [argument]

awk [-F ERE] -f *progfile* [-v assignment] [argument]

The utility ***awk*** scans each input file for lines that match any of a set of patterns specified in ***program***. Each pattern in ***program*** may have an associated action that will be performed when a line of a file matches the pattern. The set of patterns may appear either literally as ***program*** or in a file specified by using the **-f** option. This version of ***awk*** provides capabilities that were not available with previous versions. ***awk*** accepts the following options and arguments:

-F ERE

Defines the input field separator to be the Extended Regular Expression ERE. (Currently, only Basic Regular Expressions are supported.)

-v assignment

assignments in the form **x=xvalue y=yvalu**e can be passed to **awk**; **x** and **y** are awk built-in variables. The specified variable assignment occurs prior to executing the awk program, including the actions associated with ***BEGIN*** patterns, if any. Multiple occurrences of the **-v** option can be specified.

-f *progfile*

File that contains the set of pattern-action statements. If multiple instances of this option are specified, the concatenation of the files specified as ***progfile*** is used as the ***awk*** program.

program

Set of patterns for which ***awk*** scans file. To protect the program string from the shell enclose it in single quotation marks.

argument

Either of the following types of arguments can be specified:

assignment

assignments in the form **x=xvalue y=yvalue** can be passed to ***awk***; **x** and **y** are ***awk*** built-in variables.

file

Input files; if no files exist, the standard input is read. The ***file name -*** specifies the standard input. Each input line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

Redirection of Input and Output

Standard input and output may be redirected for any process with the use of the symbols < and >.

The symbol < redirects standard input (by default, the keyboard) so that input is taken from the file named after the symbol.

The symbol > redirects standard output (by default, the screen) to the file named after the symbol.

Note: ***Any information currently stored in the file specified is overwritten and lost.***

For example,

cat > quasar

will take input from the keyboard and place it in the file **quasar** until a CTRL-D is issued.

The special symbol >> appends new data to the named file.

The symbol <<***xxx*** redirects input until it encounters a line with only ***xxx***, beginning in column 1.

Pipes, Tees, and Filters

The standard output of a UNIX command can be fed directly into the standard input of another. This is the concept of the **pipe** (symbol **|**). Tapping into the stream as it proceeds from the standard output of one command to the standard input of the next is the concept of the **tee** (symbol **tee**).

A **filter** is a particular type of command that may stand in the middle positions of a series of commands linked by pipes and tees. All intermediate files necessary to the processes are handled automatically. To illustrate these concepts:

The output of **who** may be piped to the input of **pr**, in order to format it differently, and then to **wc**, in order to count the lines. The command **pr** is a filter, which formats and prints files to standard output.

```
who |  pr | wc -l
```

In order to tap into the stream between **who** and **pr**, a tee is inserted. The output of **who** will go to the file **whom** and also to the input of **pr**.

```
who | tee whom | pr | wc -l
```

Examples of filters are **cat**, **grep**, **nroff**, **pr**, **rev**, **sed**, **sort**, **tail**, **tr**, **uniq**, and **wc**, some of which are described here.

Process Status

The command **ps** will allow one to monitor the progress of processes on the system. If one has programs or other processes running in the background or in batch mode, useful information such as the amount of CPU time used may be obtained. The process identification number, the terminal initiating the command, the CPU time expended, and the command which initiated the process are all given.

If the option **-f** is selected, the status, priority, and size of the processes also are listed.

Information about the status of the NQS batch queues can be obtained using the **qstat** and **nqslimits** commands. See the online **man** pages for details.

Shell Scripts

*A shell script is a file that contains one or more UNIX commands. New shells can be created by users issuing the **sh** command.*

A shell is the process which interacts with the commands issued by a logged-on user. Each logged-on user has his own shell. A new shell (and even multiple new shells) can be created by a user with the command **sh**. The command can also execute shell scripts.

If the file **check** contained the lines

```
#
date
who
```

then the command **sh check** would create a separate shell from the one the user is operating under, execute the commands (printing the date and the current users on the screen), and return to the user's original shell. The new shell would disappear as soon as the commands were finished executing.

The command **sh** alone, without any arguments, creates new shells that the user can treat as if they were his original shell. Each is a separate environment in which commands may be issued and new processes launched. **sh** may be given repeatedly, a new shell being created each time. The command CTRL-D returns the user to the previous shell, a series of CTRL-Ds returns the user to the original sign-on shell. Issuing one more CTRL-D signs off the user .

Sample Interactive UNICOS Session

The following section demonstrates the use of some of the basic UNICOS commands on the Cray SV1.

```
% telnet asnsv1.asc.edu
Trying...
Connected to asnsv1.asc.edu.
Escape character is '^]'.

Cray UNICOS (asnsv1) (ttyp026)


login: asndsc01
Password:
Last successful login was : Wed Aug  8 15:01:16 from 129.66.32.250

        ***************************************************************
        *          Welcome to the Alabama Research and Education Network     *
        *                            UNICOS 8.0.3                            *
        ***************************************************************


                    This is a State of Alabama computer facility managed by
                    the Alabama Supercomputer Authority and its contractor
                    Nichols Research Corporation.  The system is for use by
                    State of Alabama educational institutions and authorized
                    commercial accounts.

                    In accessing the system, users consent to the monitoring
                    of usage for purposes of accounting and the detection of
                    unauthorized access.  Please note that access to this facility
                    must be specifically authorized by ASA and that unless you are
                    so authorized, access and any other use may expose you to
                    criminal and/or civil proceedings.


***********************************************************************
                                   Messages may appear here

***********************************************************************
```

```
You have mail.

asnsv1$ who                        <----- command to find out who is logged on
asnegb01        ttyp000         Aug  8 14:21  (EricB.TUCC.UAB.EDU)
root            console         Aug  7 09:55
usawaw01        ttyp001         Aug  8 13:34  (192.245.222.13)
asnphd01        ttyp002         Aug  8 14:42  (asnfddi.asc.edu)
uahycz01        ttyp003         Aug  8 13:36  (cmmr.uah.edu)
aubmlm01        ttyp004         Aug  8 12:41  (mckee.chem.auburn.edu)
asnphd01        ttyp005         Aug  8 11:54  (nrcsgi1.asc.edu)
uabgxt01        ttyp006         Aug  8 14:46  (picasso.chem.uab.edu)
oper            ttyp008         Aug  7 13:44  (129.66.32.212)
asnphd01        ttyp009         Aug  8 15:00  (asnsv1.asc.edu)
asnegb01        ttyp010         Aug  8 08:29  (EricB.TUCC.UAB.EDU)
asndlm01        ttyp014         Aug  8 10:10  (129.66.32.225)
asnjdb01        ttyp015         Aug  8 15:01  (becker.csc.usouthal.edu)
crirxh01        ttyp016         Aug  8 08:49  (cri.asc.edu)
uabtxb01        ttyp017         Aug  8 11:16  (138.26.83.1)
asnakm01        ttyp018         Aug  7 12:22  (columbia.asnvis.ua.edu)
uahycz01        ttyp020         Aug  8 14:56  (cmmr.uah.edu)
crirxh01        ttyp021         Aug  8 10:18  (cri.asc.edu)
oper            ttyp022         Aug  8 11:08  (129.66.32.212)
aamgsl01        ttyp023         Aug  8 11:48  (198.180.132.250)
uabsck01        ttyp024         Aug  8 15:00  (ftp.PHY.UAB.EDU)
aubfxs01        ttyp025         Aug  8 13:28  (kharazmi.eng.auburn.edu)
asndsc01        ttyp026         Aug  8 15:02  (asnmail.asc.edu)
ualhxk01        ttyp029         Aug  8 14:07  (uarrp.gri.ua.edu)
uahymc01        ttyp030         Aug  8 14:17  (ebs330.eb.uah.edu)


asnsv1$ finger paul        <----- command to identify a user or users id
Login name: asnphd01                  In real life: Paul H Duggan
Directory: /home/staff/asnphd01         Shell: /bin/csh
On since Aug  9 08:48:55 on ttyp020 from nrcsgi1.asc.edu
1 hour 7 minutes Idle Time
No Plan.

Login name: uahpmd01                  In real life: Paul M. Dean
Directory: /home/uah/uahpmd01           Shell: /bin/csh
Last login Thu Feb 23 08:13 on ftpd from ebs330.eb.uah.edu
No Plan.

Login name: usapny01                  In real life: Paul N. Younger
Directory: /home/usa/usapny01           Shell: /bin/csh
Last login Thu May 26, 1995 on /dev/ttyp002 from usaaxp.usa.edu
No Plan.

Login name: uabpko01                  In real life: Paul K. Owens
```

```
Directory: /home/uab/uabpko01          Shell: /bin/csh
Last login Mon Sept 16 17:11 on /dev/ttyp011 from display.asnvis.uab.edu
No Plan.



asnsv1$ ls -l                     <----- command to list files
total 2520
-rwx------     1 asndsc01 staff       214840     Aug  8 14:58 a.out
-rw-r-----     1 asndsc01 staff       278        Aug  8 14:00 c94.job
-rw-r-----     1 root     staff       1993       Aug  8 13:46 c94.job.o38160
drwx------     2 asndsc01 staff       4096       Aug  8 11:57 hello
-rw-------     1 asndsc01 staff       2011       Aug  8 13:56 hello.out
-rw-------     1 asndsc01 staff       77         Aug  8 14:58 name.c
-rw-------     1 asndsc01 staff       75         Aug  8 14:51 name.p
drwxr-xr-x     2 asndsc01 staff       4096       Aug  3 09:55 ncube

asnsv1$ cd hello                  <----- command to change directories


asnsv1$ pwd                       <----- command to print working directory
/usr/staff/asndsc01/hello

asnsv1$ ls -l               <----- command to list files
total 2008
-rwx------      1 asndsc01 staff      1023960    Aug  8 11:57 a.out
-rw-r-----      1 asndsc01 staff      75         Aug  8 11:50 hello.f

asnsv1$ cat hello.f         <----- command to list contents of file hello.f
      program hi
      print*, 'Hello '
      stop
      end

asnsv1$ cat hello.f >hello.newf         <----- list hello.f to hello.newf

asnsv1$ cat hello.newf            <----- list hello.newf
      program hi
      print*, 'Hello '
      stop
      end

asnsv1$ mkdir fortran             <----- command to make a directory

asnsv1$ ls -l               <----- list files
```

```
total 2024
-rwx------      1 asndsc01 staff        1023960     Aug  8 11:57 a.out
drwx------      2 asndsc01 staff        4096        Aug  8 15:45 fortran
-rw-r-----      1 asndsc01 staff        75          Aug  8 11:50 hello.f
-rw-------      1 asndsc01 staff        75          Aug  8 15:45 hello.newf

asnsv1$ cd fortran              <----- change to directory named fortran

asnsv1$ ls -l              <----- list files
total 0                    <----- there are no files in this directory at this time
asnsv1$ pwd                     <----- print working directory
/usr/staff/asndsc01/hello/fortran

asnsv1$ cp /usr/staff/asndsc01/hello/hello.f hello.ff      <----- copy file hello.f from
                                                    parent directory to hello.ff in
                                               new  directory
asnsv1$ ls                      <----- list file to see if hello.ff was copied to the new
                                       directory
hello.ff

asnsv1$ cat hello.ff            <----- list contents of hello.ff
      program hi
      print*, 'Hello '
      stop
      end

asnsv1$ date                    <----- prints current date on screen
Wed Aug  8 15:49:10 CST 1995

asnsv1$ ls -l              <----- list files
total 8
-rw-r-----      1 asndsc01 staff        75          Aug  8 15:48 hello.ff

asnsv1$ chmod u+x hello.ff      <----- change permissions on hello.ff

asnsv1$ ls -l              <----- list file to see new permissions
total 8
-rwxr-----   1 asndsc01 staff        75 Aug  8 15:48 hello.ff

asnsv1$ cd ..                   <----- change to parent directory
```

```
asnsv1$ pwd                    <----- print to check name of parent directory
/usr/staff/asndsc01/hello

asnsv1$ rmdir fortran          <----- command  to delete a directory
rmdir: fortran -- Directory not empty

asnsv1$ cd fortran             <----- change to directory named fortran to delete
                                 contents

asnsv1$ ls                     <----- list files
hello.ff

asnsv1$ rm hello.ff            <----- command to remove the file named hello.ff

asnsv1$ cd ..                  <----- change to parent directory

asnsv1$ rmdir fortran          <----- remove or delete directory named fortran

asnsv1$ ls -l             <----- list file to see if fortran directory was deleted
total 2016
-rwx------     1 asndsc01 staff        1023960      Aug  8 11:57 a.out
-rw-r-----     1 asndsc01 staff        75           Aug  8 11:50 hello.f
-rw-------     1 asndsc01 staff        75           Aug  8 15:45 hello.newf

asnsv1$ exit              <----- command to log off or end session on Cray
Connection closed.
%
```

Introduction to Line-Oriented Text Editor ex

To invoke **ex** type the following command:

***ex** [options] filename*

(**ex** can also be accessed from within **vi**, the visual editor; See the description below of **vi**.) The options are as follows:

*The **ex** text editor is powerful and useful in an environment in which the use of a full screen editor is not practical.*

(Reproduced with permission of the publisher, Howard W. Sams and Co., Indianapolis Indiana, UNIX System V Primer, Waite, Augtin and Prata, © 1984.)

-	A minus suppresses messages that would be sent by the editor during an interactive session.
-v	Calls up the **vi** editor.
-t ***tag***	Allows one to edit the file containing the tag, positioning the editor at the tag.
-r ***file***	Recovers the named file after a system crash. If no file is specified, a list of recoverable files is given.
-R	Allows the file to be read only locking out accidental editing.
+***command***	Editing begins immediately with the command given.
-L	Begin LISP mode. LISP programs will be properly aligned.

*The **ex** editor has three different modes: the Command, the Insert, and the Visual Modes. The command mode is the normal mode in which one enters the editor.*

-X Begin encryption mode. (Edit an encrypted file.)

In the **command mode**, the editor prompts with a colon for the commands summarized under command mode. This is the normal mode in which one enters the editor.

The **text input or insert mode** is invoked with one of three commands **a**, **i**, or **c**, as summarized under Text Input Mode.

The **visual mode,** invoked with **vi**, enters **vi**; **ex** is reentered by typing **Q** or **e** in **vi**'s command mode.

Sample Session

The user should be in the directory in which the new file is to be located.

To begin editing a new file called **test** in the current working directory, type:

```
$ ex test
```

The editor will respond with

```
"test"[Newfile]
:
```

To insert text, use the append command, **a,** at the colon prompt**:**

```
:a
Fourscore and seven years ago,
our forefathers brought forth
on this continent  (etc)

guv'ment for the people,
of the people, by the people
shall not perish  (etc).
```

New text will be appended to the current line position (in this case the top of the file) until a single period and only a single period, is entered on a line. Then insert mode is terminated, and the colon prompt signifying command mode is returned. Text can be saved by writing it with the **w** command and exiting the editor with the **q** command.

The user can now edit **test** and change some of the text. Enter

```
$ ex test
(editor responds with filename and
number of lines
and characters in the file)
:
```

Move to the sixth line of the file by giving four carriage returns or by having the editor print line 5 (**5p**). Then use the substitute command to change **guv'ment** to **government:**

```
:5p
guv'ment for the people
:s/guv'ment/government/
government for the people
:
```

Note that the line is displayed after the substitution. The change could be saved with a **w** command.

Text Input Mode

a **A**ppend lines after current line, unless line number is specified; e.g. **3a** adds new text after line 3.

i **I**nsert lines before current line, unless line number is specified; e. g. **3i** adds new text before line 3.

c **C**hange current line or lines specified in address; e. g. **2,4c** deletes lines 2 through 4 and adds new text that is typed in.

Command Mode

p	**P**rint on the screen specified lines. If no lines are specified, print the current line. **2,4p** prints lines 2, 3, and 4.
d	**D**elete specified lines. If none specified, delete line dot; e. g. **5,8d** deletes lines 5 through 8.
m	**M**ove specified lines to line named after **m**; e. g. **1,2m5** moves lines 1 and 2, placing them after line 5.
co	**Co**py specified lines to line named after **co**; e. g. **2,4co$** copies lines 2 through 4, placing them at the end of buffer.
r ***filename***	**R**ead in the contents of ***filename*** at current line or at line specified.
S*/one/two/*	**S**ubstitute the word *"two"* for the word *"one"* for the first occurrence in the specified lines.
/*nice*/	Search for the next line to contain the word(s) between the slashes: in this case, the string "*nice*".
g	**G**lobal search or substitute generally used with **s** or **s/pat1/pat2/g**. Substitutes "pat2" for "pat1" for all occurrences of "pat 1" in the specified lines.
nu	**Nu**mber the lines and print them on the screen; e.g..**1,$nu** numbers and prints all the lines in a file.
u	**U**ndo the last change made in the buffer.
z	Print lines on the screen; e. g., **3z** prints the file on screen starting with line 3.

Leaving the editor

w	**W**rite the specified lines that are addressed to a named file; e. g., **2,5w popcorn** writes lines 2 through 5 into file **popcorn**.
q	**Q**uit.
q!	**Q**uit without writing changes to file.

There are many good books with tutorials on the use of the editor ***ex.*** *One such is published by Cray Research, Inc. and is titled UNICOS Text Editors Primer (publication number SG-2050).*

Addressing Lines

.	This character addresses the current line, called "line dot". The current line is the last line affected by a command. Thus, .p prints the current line.
.=	This command prints the line number of the current line; e. g., editor responds with a number like "5".
$	This character addresses the last line in the buffer; e. g., $d deletes the last line.
n	A decimal number n addresses the nth line; e. g., 3p prints line number 3.
+ -	The + and - are used in conjunction with a reference line which may be specified with n or $ or current line; e. g., $-5,$p prints the last six lines of the buffer.
<return>	When used with no command, it is equivalent to the next line. Very useful for stepping through the buffer.

Note: *If users accidentally hit* ***o*** *or* ***vi*** *while in* ***ex****, the way to get back to* ***ex*** *is to hit the* ***<esc>*** *key, then the* ***Q*** *key.*

Introduction to the Screen-Oriented Editor vi

*The visual editor **vi** must be customized to a particular visual terminal. **vi** cannot be used with an IBM 3270 terminal.*

(Reproduced with permission of the publisher, Howard W. Sams and Co., Indianapolis Indiana, UNIX System V Primer, Waite, Augtin and Prata, © 1984.)

Sample vi Session

Edit a file in the directory where the file is located or should be located (in the case of a new file). Begin editing a new file by typing the following:

$ vi *file*

Insertion of text can be done after issuing the **i** command:

```
i
Augy had a little lamb,
its fleece was white as snow (etc)
<esc>
```

Move about in the file with the **h,j,k,and l** keys when in command mode (the <esc> returns one to command mode).

Cursor Positioning Keys--in the Command Mode

<h>	Moves cursor one character to the left.
<j>	Moves cursor down one line anywhere in text.
<k>	Moves cursor up one line anywhere in text.
<l>	Moves cursor one character to the right.

Move with these keys to the "f" in "fleece" in the second line, and type **cw.** The word "fleece" disappears, replaced by a $. Type "fur" in substitution.

To save the file, with changes, type **<esc> :ZZ**. This exits to the shell.

Entering text input mode--End this mode with an <esc>

a	**A**ppend text after the cursor. Enter as many lines and<return>'s as needed.
i	**I**nsert text before the cursor. Enter as many lines of text and <return>'s as needed.
o	**O**pen a new line below cursor. Ready for the text input.
O	**O**pen a new line above cursor. Ready for the text input.
R	**R**eplace characters on the screen, starting at the cursor, with any characters typed.

These commands, after execution, return the editor to the command mode

r	**R**eplace a single character under the cursor with a single character that is typed.
/***happy***	Search sequence; looks for next occurrence of pattern following / (in this case, the word "*happ*y").
?*lark*	Search sequence; like /, but searches backwards from the cursor.
n	Used after / or **?** to advance to the next occurrence in the buffer of the pattern.
u	**U**ndo the last command.

U	**U**ndo all the changes to the current line.
x	Delete character highlighted or underlined by the cursor.

<del> or # or <CTRL-H> or <rub>
This backspace feature of the shell also works in the editor. These commands move the cursor character by character, left within a line, erasing each character from the buffer.

<CTRL-F>	Scroll or page the screen forward one page at a time.
<CTRL-B>	Scroll or page the screen backward one page at a time.
<CTRL-D>	Scroll or page the screen down one-half page at time.
<CTRL-U>	Scroll or page the screen up one-half page at time
<CTRL-G>	Identify the line where the cursor is located by line number.
nG	Position the cursor at line **n** in the file.

Operators in the Command Mode

d	**D**elete indicated text starting at the cursor. For example, use **dw** to delete a word and **dd** to delete a line; **3dd** deletes 3 lines. Deleted text is stored temporarily in a buffer whose contents can be printed out with the **p** command. Also, **d** can be used with named buffers in the manner described for **y** command.

c	Delete indicated text starting at the cursor and enters Text Input Mode. Thus, **cw** deletes from the cursor to the **end** of the word, allowing users to add text between those positions.
y	Copy indicated text, starting at the cursor, and stores it in a buffer. There are nine unnamed buffers (1-9) that store the last nine **d**elete or **y**ank operations, and 26 named buffers (a-z) that can be used for storage. The double quote mark (") is used to tell the editor the name of the buffer. Thus, **"cy$** will store text from the cursor to the end of the line in a buffer named **c**.
p	Inserts "delete" and "yank" buffer contents after the cursor or on the next line. Command **p** puts the last item yanked or deleted back into the file just after the cursor, and **"cp** will put the contents of buffer **c** after the cursor.

Scopes for Use with Operators

e	The scope from the cursor to the **end** of the current word; e. g., if the cursor is on the "u" in "current", and the user types **de**, then "urrent" is deleted.
w	The scope is from the cursor to the beginning of the next **word**, including the space.
b	The scope is from the letter before the cursor, backwards, to the **beginning** of the word.
$	The scope is from the cursor to the end of the line.
0	The scope is from just before the cursor to the beginning of the line.

)	The scope is from the cursor to the beginning of the next sentence. A sentence is ended by ".", "!", or "?", followed by 2 spaces or by an "end of line" (provided by the <return> key).
(	The scope is from just before the cursor back to the beginning of the sentence containing the cursor.
}	The scope is from the cursor to the end of a paragraph. A paragraph begins after an empty line.
{	The scope is from just before the cursor back to the beginning of a paragraph.

Leaving the Editor

<esc>:w	Write the contents of the buffer into the current file of the same name. Can write to a new filename. Also, can send partial buffer contents using line numbers, such as **A:3,10w popcorn**.
<esc>:q	Quit the buffer after a :**w** command.
<esc>:wq	Write and quit, placing buffer contents in file.
<esc>:q!	Quit editor without making changes in file. ***Dangerous***.
<esc>ZZ	Write and quit, placing buffer contents in file.

Using the ex editor while in vi

:	Generate a colon (:) prompt at the bottom of the screen and let users make one **ex** command. Users are returned to the **vi** mode when the command finishes execution.
Q	**Q**uit **vi** and place users in the **ex** editor, giving users a command mode prompt, the colon (:) at the bottom of the screen. Users can get back to **vi** while in the command mode.

When in Doubt

<esc>	Put users in the command mode.

There are many good books with tutorials on the use of the editor vi. One such is published by Cray Research, Inc. and is titled UNICOS Text Editors Primer (publication number SG-2050).

Compiling and Running Programs

There are three compilers available on the ASC Cray SV1.

C (cc)
C++ (CC)
FORTRAN 90 (f90)

This section provides examples on running programs using the F90 and C compilers. Batch processing uses the same commands embedded in the batch file.

FORTRAN Programs

The **f90** is a full ANSI FORTRAN 90 compiler, with enhanced optimization and vectorization capabilities; **f90** is the recommended compiler for the majority of users. **f90** is the newly-developed FORTRAN autotasking compiler which can automatically optimize FORTRAN code for multi-processor Cray systems; **f90** offers a variety of options for experienced users.

The user should check the manual for precise use of the various options. Online documentation is available with the **man f90** command.

The sequence for running a FORTRAN program is as follows:

```
asnsv1$f90 source.f
```

*source must be in a **.f** file*

```
asnsv1$./a.out
```

*f90 produces **a.out** executable*

Example of Compiling and Running a Program using f90

```
asnsv1$ ls
hello.f
asnsv1$ f90 hello.f
asnsv1$ ls -l

total 2016
-rwx------   1 asndsc01 staff    1023960 Aug  9 15:16 a.out
-rw-r-----   1 asndsc01 staff         ......75 Aug  9 11:50 hello.f

asnsv1$ ./a.out
 Hello from the SV1!
 STOP executed at line 3 in Fortran routine 'HI'
 CP: 0.002s,  Wallclock: 0.035s,  2.2% of 2-CPU Machine
 HWM mem: 131987, HWM stack: 2048, Stack overflows: 0
asnsv1$
```

The default executable file is ***a.out***. This can be changed with the **-o** option:

```
asnsv1$f90  -o  myexec.hello hello.f

asnsv1$./myexec.hello
```

The listing file is controlled by the **-e** and **-r** options. The default is no listing. The form is:

```
asnsv1$f90  -e options  -r listfile hello.f
```

-e controls the options to be turned on;   **-d** controls the options to be turned off.

C Programs

*The Cray C++ compiler is invoked with the **CC** command. It has enhanced optimization and vectorization features and complies with the ANSI-C standard.*

Online documentation is available with the **man cc** command.

The C compiler is called with the **cc** command:

```
  asnsv1$cc source.c
          compile source.c,
          produces  executable a.out

  asnsv1$./a.out     <=== execute the program
```

It is not necessary to call the segldr if the C compiler is called without options and with a single input file.

Example of Listing, Compiling, and Running a C program

```
asnsv1$ ls
hello.c
asnsv1$ ls -l
total 8
-rw-------      1 asndsc01 staff        77          Aug  9 15:40 hello.c
asnsv1$ cat hello.c
#include <stdio.h>
main(){
printf ("Hello World .....from C\n");
return;
}


asnsv1$ cc hello.c
asnsv1$ ls -l
total 432
-rwx------      1 asndsc01 staff        214840        Aug  9 15:40 a.out
-rw-------      1 asndsc01 staff        77            Aug  9 15:40 hello.c
asnsv1$ ./a.out
Hello World .....from C
asnsv1$
```

Preparing a Batch Job

A batch job is essentially a shell script file that is submitted via NQS to the batch queues.

A sample batch script follows:

```
asnsv1$ cat xmp.job

#!bin/sh
set -x <-- echo statements to
           output

cd /tmp/asnger08
cd plots
f90 fplot.f
#
# end batch script
#
```

File-Naming Precautions

When users are running multiple batch jobs, they must ensure that file names are unique.

Under UNICOS, a user's files are shared by all batch and interactive jobs running under the same userid. Hence two separate jobs running simultaneously under the same userid, each writing to file abc, will write to the same file and produce garbage. One way to avoid doing this is to qualify file names by the job process id, which is accessed through the shell variable ***$$***.

For example, instead of abc.xyz, use **abc.$$.xyz**.

Another method, illustrated below, is to create a directory qualified by the process id, and place all temporary files in that directory.

Files created by batch jobs are **NOT** automatically removed at job termination unless they are located in the **/tmp** directory. Any files in **/tmp** are removed upon exiting a session. Files not in **/tmp** stay in the created working directory until explicitly deleted with the **rm -r** command.

Submitting Batch Jobs

UNICOS provides a facility called NQS (Network Queuing System) for the submittal of 'batch' jobs. A batch job is a standard UNICOS command stream.

Batch jobs may be submitted using the **qsub** command. For example:

```
asnsv1$ qsub -q express myscript
```

```
asnsv1$ qsub -lm 12mw -lt 600 myscript
```

Using qsub

The **qsub** command is issued by a user to submit a UNICOS file into the batch execution queue. By default, all outputs - **stdout and stderr** - remain on the SV1 as files ***filename.oprocess*** and ***filename.eprocess.*** However, the Bourne shell variable **$ROUT** can be defined to ensure that such output is returned to the remote node. For complete information on qsub, see **man qsub**. The format of the qsub command is:

```
qsub [options] file
```

options

There are various options to control the batch job. Most of these options can also be included in the batch job stream as **#QSUB** option.

Some important options are:

-eo
Directs stderr output to the same destination as stdout.

-o filename
Directs stdout to the stated file destination.

Note:
By default, under NQS all stdout and stderr I/O is spooled into a private NQS directory. The output files are returned ***upon job completion****.*

The user should redirect the job I/O at run-time in order to prevent being unable to view any results until job completion. To redirect job I/O:

./a.out > /tmp/stdout.$$ 2>stderr.$$

-lt cputime
Set process time limit to cputime CPU seconds.

-lm memsize
Set process memory limit to memsize.

file
The UNICOS command file to submit to the batch queue.

Example:
A job file SV1.job is submitted to the **express** queue by choosing memory and time limits associated with the **express** queue and directing both **stderr** and **stdout** to the same output file (**-eo** option):

```
asnsv1$qsub -eo -lt 500 -lm 5mw -q batch SV1.job
Request 8602.asnSV1 submitted to queue: express.
asnsv1$

asnsv1$qstat -r
------------------------------
NQS 61.2 BATCH REQUEST SUMMARY
------------------------------
IDENTIFIER    NAME    USER     QUEUE                JID  PRTY REQMEM REQTIM ST
------------ ------- -------- -------------------- ---- ---- ------ ------ ---
 8602.asncray SV1.job asnjsn01 express              2357   15   5000    600 R03
no pipe queue entries
no device queue entries
asnSV1$
```

Output shows up in file SV1.job.o<pid>

```
asnsv1$
asnsv1$ls -l xmp*
-rw-r--r--   1 asnjsn01 root          79 Jun 14 18:03 SV1.job
-rw-r--r--   1 asnjsn01 root        5569 Jun 14 18:03 SV1.job.o8602

asnsv1$cat SV1.job.o8602

[.....message of the day.........}
You have mail.
news: gaussian CF7750.news amber g92

Mon Jun 3 18:03:16 CDT 1996
```

```
+ cd /tmp/asnjsn08
+ cd plots
+ f90 fplot.f

            [ output from the job shows up here ]

+ test -n
+ test -n BATCH
+ test BATCH = BATCH
asnsv1$
```

Selecting a Job Priority

Job Priority Selections:

Express
1.4 times batch cost

Prime
1.0 times batch cost
(default)

Non-Prime
0.6 times batch cost

Jobs submitted to the express queue are automatically charged at 1.4 times the prime time rate. Interactive and batch jobs that run between 8:00 a.m. and 6:00 p.m. are charged at the prime rate. Interactive sessions and batch jobs that run between 6:00 p.m. and 8:00 a.m. are charged at the non-prime rate.

The **nqslimits** command will display a list of the available NQS queues, their respective priorities, and limits for CPU time, memory usage and file size.

ftp (File Transfer Protocol)

Files can be transferred between systems on the network using the **ftp** command. Files are usually transferred as ASCII files. Binary files can also be transferred. Use of **ftp** requires a valid userid and password on the remote system. The commands covered in this section are:

ftp	Establish a remote connection.
get	Move a file from the remote host to the local host.
put	Move a file from the local host to a remote host.
mget	Retrieve multiple files from a remote host.
mput	Send several files to a remote host.
mkdir	Create a new directory on a remote host.
cd	Change directories on a remote host.
dir	List files in the current remote directory.
quit	Exit from ftp.

For additional information about the ftp command, enter **man ftp**.

Connecting to a Remote System

To establish a connection to a remote system, use the **ftp** command. After the connection is established, provide a valid userid and password:

```
>> ftp asnsv1.asc.edu
Connected to asnsv1.asc.edu.
220  asnsv1  FTP  server  (Version  5.2  Fri
Sep 7 14:09:58 CDT 1995) ready.
Name (asnsv1.asc.edu:asnxyz01): asnxyz01
331 Password required for asnxyz01.
Password:
230 User asnxyz01 logged in.
ftp>
```

The get Command

The **get** command is used to transfer a file from the remote system to the local system. The syntax is:

get *remotefilename localfilename*

If the local file name is omitted, the remote file name will be used for both files.

```
ftp> get file1.x
200 PORT command successful.
150  Opening  ASCII  mode  data  connection
for file1.x (14 bytes).
226 Transfer complete.
local: file1.x remote: file1.x
15  bytes  received  in  0.04  seconds  (0.37
Kbytes/s)
ftp>
```

The put Command

The **put** command is used to send a file from the local host to the remote host. The syntax is:

put *localfilename remotefilename*

If the remote file name is omitted, the local file name will be used for both files.

```
ftp> put xyz.x file3.x
200 PORT command successful.
150 Opening ASCII mode data connection
for file3.x.
226 Transfer complete.
local: xyz.x remote: file3.x
15 bytes sent in 1e-06 seconds (1.5e+04
Kbytes/s)
ftp>
```

The mget Command

The **mget** command is used to transfer multiple files from the remote host to the local host. Standard UNIX 'wildcard' characters can be used. The syntax is:

mget *wildcardname*

for example ***mget file**** will transfer all files beginning with the characters 'file'. The mput and mget commands typically require confirmation of every file unless ftp is invoked with a “-i" option.

```
ftp> mget file*
200 PORT command successful.
150 Opening ASCII mode data connection
for file1.x (14 bytes).
226 Transfer complete.
local: file1.x remote: file1.x
15 bytes received in 0.04 seconds (0.37
Kbytes/s)
200 PORT command successful.
150 Opening ASCII mode data connection
for file2.x (14 bytes).
226 Transfer complete.
local: file2.x remote: file2.x
15 bytes received in 0.05 seconds (0.29
Kbytes/s)
ftp>
```

The mput Command

The **mput** command is used to transfer multiple files from the local host to the remote host. Standard UNIX 'wildcard' characters can be used. The syntax is:

mput *wildcardname*

For example **mput file*** will transfer all files beginning with the characters 'file'.

```
ftp> mput file*
200 PORT command successful.
150 Opening ASCII mode data connection
for file1.x.
226 Transfer complete.
local: file1.x remote: file1.x
15 bytes sent in 1e-06 seconds (1.5e+04
Kbytes/s)
200 PORT command successful.
150 Opening ASCII mode data connection
for file2.x.
226 Transfer complete.
local: file2.x remote: file2.x
15 bytes sent in 1e-06 seconds (1.5e+04
Kbytes/s)
ftp>
```

The mkdir Command

The **mkdir** command is used to create a new subdirectory on the remote host. The syntax is:

mkdir *new-sub-dir*

```
ftp> mkdir demoftp
257 MKD command successful.
ftp>
```

The cd Command

The **cd** command is used to change directories on the remote host. The syntax is:

cd *dir-name*

```
ftp> cd demoftp
250 CWD command successful.
ftp>
```

The dir Command

The **dir** command is used to get a listing of the files in the current remote directory. The syntax is:

dir

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /usr/ucb/ls.
total 8
-rw-r-----  1 asnger01 u_staff        14 Jun 13 15:58 file1.x
-rw-r-----  1 asnger01 u_staff        14 Jun 13 15:58 file2.x
226 Transfer complete.
135 bytes received in 0.13 seconds (1 Kbytes/s)
ftp>
```

The quit Command

The **quit** command exits from the **ftp** session. The syntax is:

quit

```
ftp> quit
221 Goodbye.
>>
```

OPTIMIZATION

This chapter provides information about techniques you can use to optimize Fortran, C, or C++ code on the AREN Cray SV1 supercomputer.

Optimization is the process of changing a program or the environment in which it runs to improve its performance. Performance gains generally fall into one of two categories of measured time:

- ***user* CPU *time***. Time accumulated by a user process when it is attached to a CPU and executing. When running on a single CPU, CPU time is a fraction of elapsed time. When multitasked, CPU time is a multiple of elapsed time.

- ***elapsed* (wall-clock) *time***. The amount of time that passes between the start and termination of a user process. Elapsed time includes the following:

  - User CPU time

  - UNICOS system CPU time

  - I/O wait time

  - Sleep or idle time

Optimization begins with code that has been debugged and is running on the SV1 system. The code has been compiled using the best compiler options and data types for your program. This means you have used the aggressive optimization options (-Oscalar3, vector3 for the Fortran f90 command or -hscalar3, vector3 for the C or C++ cc or CC command) and, if you do not need 96-bits of floating-point precision, you have used single precision (f90 -dp or the float or double type specifier instead of long double type specifier within C++ code).

Optimizing code on the SV1 is an iterative process requiring the following steps:

- evaluate the code
- determine possible areas where optimization techniques can be applied
- apply the techniques
- check code performance
- is code sufficiently optimized?
  - if not, return to evaluation phase
  - if yes, the code is optimized for single CPU performance

The first step in optimizing a program is evaluating its overall performance. This allows you to decide where to focus optimization efforts. When you compile and execute a program on the SV1 system, it usually will be dominated by one of the following general activities:

- Memory management
- Input/output (I/O) processing
- CPU computation

A program is considered to be **memory** bound, **I/O** bound, or **CPU** bound because its performance is limited by the dominant activity. Use the job accounting (**ja**) and hardware performance monitor (**hpm**) tools to identify where in your code to obtain the greatest potential performance gain**.**

If your evaluation shows that your code is **memory** bound, then:

- apply memory management techniques

If evaluation shows your code to be **I/O** bound, then:

- use the *ja* report to determine:

  - is there excessive unlocked wait time?
  - are the I/O requests efficient?
  - are you getting the expected I/O transfer rates?

- apply I/O optimization techniques

If your evaluation determines that your code is **CPU** bound, then:

- apply **CPU** optimization techniques for the following conditions:

  - the code is achieving low MFLOPS and MIPS rates
  - the code has a high fetch rate
  - the code has low computational intensity

The following tools are available on the SV1 to assist you in evaluating and optimizing you code.

Tool	Type	Description
cflist(1)	Command	Generates a listing of a Fortran program.
flowview(1)	Command	Displays information from the raw data generated by the Flowtrace library.
Flowtrace	Feature	Times subroutines and functions during program execution.
FLOWMARK	Subroutine	Times specified areas of programs (not necessarily subroutines or functions). FLOWMARK is a special feature of Flowtrace and Perftrace.
hpm(1)	Command	Gathers Hardware Performance Monitor (HPM) statistics for an entire program. The perfview(1) command displays data from the hpm(1) command.
ja(1)	Command	Provides accounting data about many aspects of job processing.
perfview(1)	Command	Displays information from raw data generated by the Perftrace library, or the hpm(1) command
Perftrace	Feature	Times subroutines and functions based on statistics gathered from the Hardware Performance Monitor (HPM) device. The perfview(1) command displays Perftrace data.
procstat(1)	Command	Gathers statistics about I/O, process, and memory as a program executes. The procview(1) command displays data generated by the procstat(1) command.
procview(1)	Command	Displays information from the raw data generated by the procstat(1) command.
xbrowse(1)	Command	Provides an interactive, graphic interface in which to view and edit Fortran 90, and Cray Standard C codes.

For assistance in using these tools and for help in evaluation and optimization of your code, please see your resident Computational Research Specialist.