# Table of Contents

**11 System Generation**

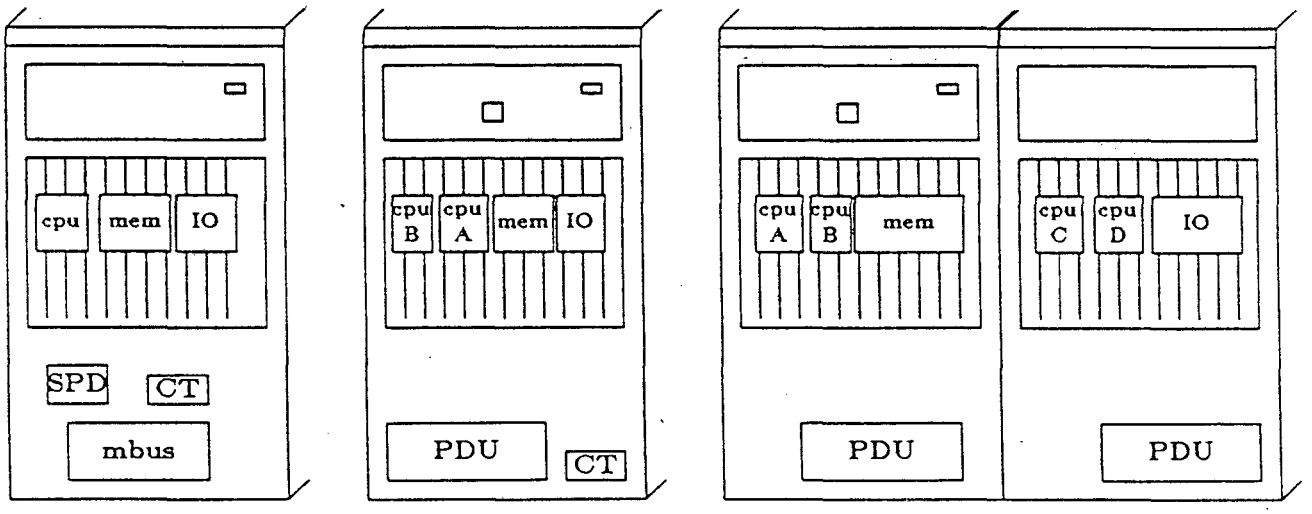


**12 Security Features**

# A FAMILY OF COMPATIBLE SUPERCOMPUTERS

| | Relative speed | Number of Processors | |
|---|---|---|---|
| C120 | 1X | 1 | Entry Level Supercomputer |
| C201 | 1.7X | 1 | |
| C202 | 3.4X | 2 | Mid Range Parallel Processor |
| C210 | 2.5X | 1 | |
| C220 | 5.0X | 2 | |
| C230 | 7.5X | 3 | |
| C240 | 10.0X | 4 | Parallel Processing Supercomputer |

## 10X Range of Compatible Performance




C120 C210,220 C230,240

# Summary of the C Series

| Processor | Relative Performance | Number of Processors | Cycle Time nsecs | Max Memory GB | Memory B/W MB/sec | I/O B/W MB/sec |
|---|---|---|---|---|---|---|
| C120 | 1 | 1 | 100 | 1 | 80 | 80 |
| C130 | 1.7 | 1 | 60 | 1 | 160 | 80 |
| C210 | 2.5 | 1 | 40 | 2 | 200 | 160 |
| C220 | 5.0 | 2 | 40 | 2 | 400 | 160 |
| C230 | 7.5 | 3 | 40 | 2 | 600 | 160 |
| C240 | 10.0 | 4 | 40 | 2 | 800 | 160 |

➤ *10X Range of Compatible Performance*

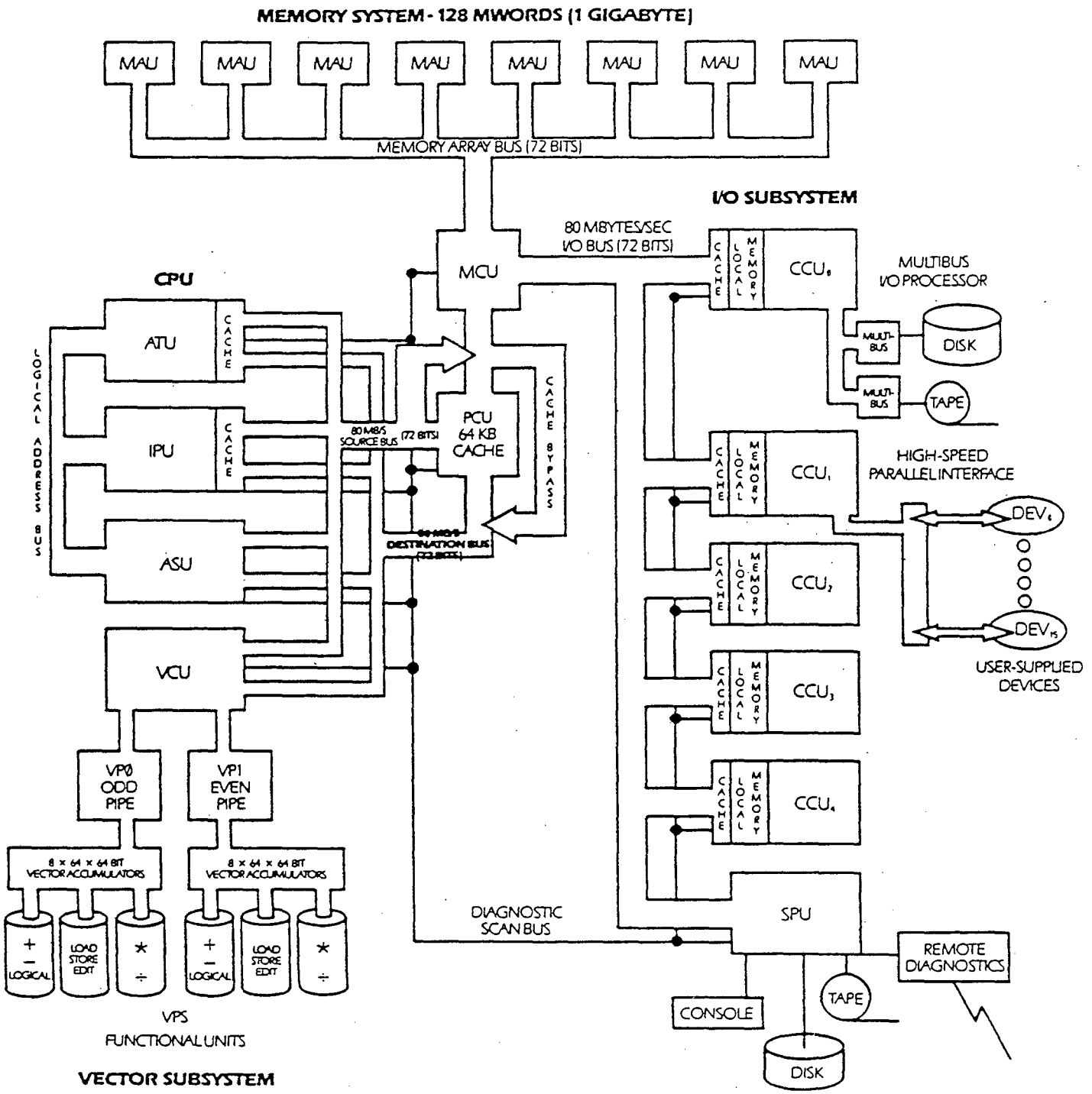
# C1 FUNCTIONAL SUBSYSTEMS

- Is a tightly coupled asymmetric multiprocessor
- Is composed of five functional subsystems:
  - o Service Processor Subsystem
  - o Input/Output Subsystem
  - o Memory Subsystem
  - o Central Processing Unit Subsystem
  - o Electromechanical Subsystem
- The system is asymmetric because each processor performs a unique set of tasks
- The system is tightly coupled because communication between the subsystems is through shared memory
- ConvexOS is the main operating system

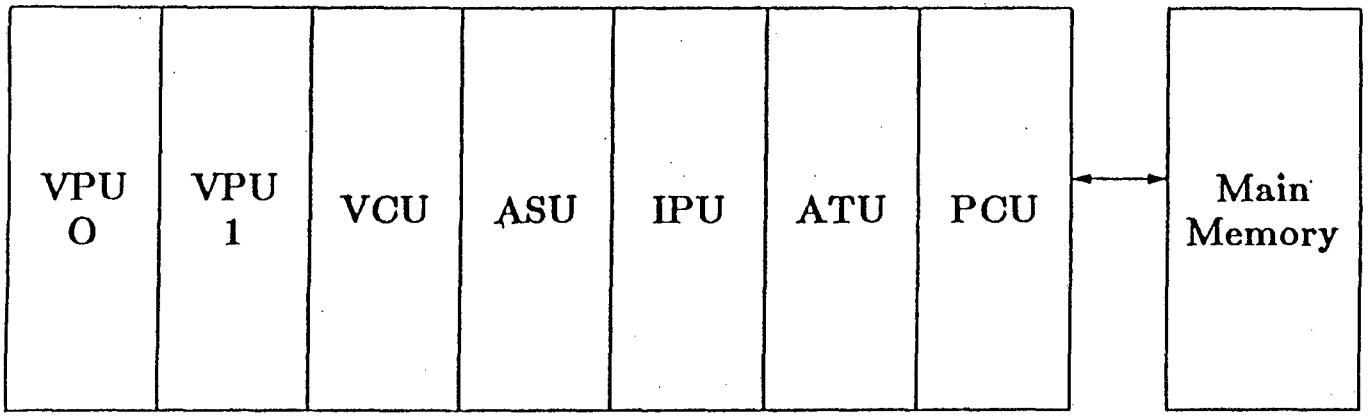
# C1 SYSTEM FEATURES

- High-speed scientific computer with integrated vector and scalar processing
- RISC (Reduced Instruction Set Computer) architecture
- Highly-pipelined implementation
- Both 32-bit and 64-bit calculations supported in hardware
- Independent, high-performance intelligent I/O subsystems
- High-speed MULTIBUS and/or VMEBUS subsystem with up to 160 MULTIBUS I/O controllers
- Extensive reliability, availability, and serviceability features
- Compact 19-inch RETMA(ruggedized extensively tested military accepted) package, air-cooled
- Rugged construction and low power consumption

# C1 ARCHITECTURE



# C-1 JOB PROCESSOR




VPU0 - Vector Processing Unit 0
VPU1 - Vector Processing Unit 1
VCU - Vector Control Unit
ASU - Address & Scalar Unit
IPU - Instruction Processing Unit
ATU - Address Translation Unit
PCU - Physical Cache Unit

# C1 MEMORY SYSTEM

- Maximum physical memory
  - o Original C1 - 128 megabytes
  - o C1 XL - 64 megabytes
  - o C1 XP - 1 gigabyte
- Virtual address space - 4 gigabytes
- Maximum user program space - 2 gigabytes
- Memory page size - 4096 bytes
- Memory bandwidth - 80 megabytes/second
- Memory interleaving
  - o Original C1 - fixed, four-way
  - o C1 XL - fixed, four-way interleaving
  - o C1 XP - extended interleaving (from four-way to 32-way)
- Physical cache - 64 kbytes; 50 ns access
- Instruction cache
  - o C1 XL - 1024 bytes
  - o C1 XP - 4096 bytes

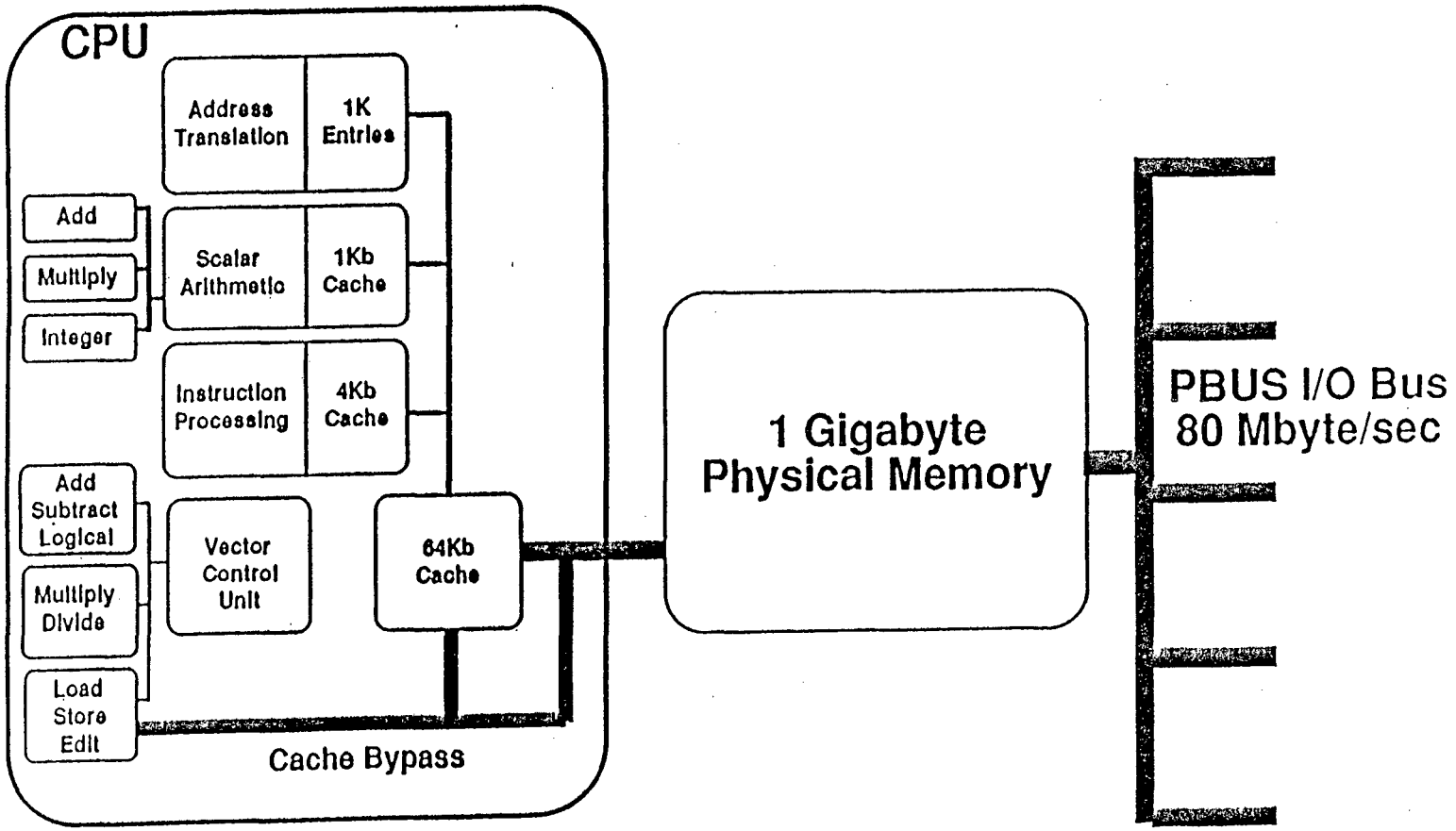
# C1 REGISTERS

- Data paths are 72 bits wide (64 data bits plus 8 SECDED bits)
- Address registers (A regs)
  - o 32 bits wide
  - o Eight registers (A0 - A7)
- Scalar data registers (S regs)
  - o 64 bits wide
  - o Eight registers (S0 - S7)
- Vector registers (V0 - V7)
  - o 64 bits wide by 128 element deep
  - o Eight registers (V0 - V7)
- Processor cycle time - 100 nanoseconds
- Peak processing
  - o 40 MFLOPS (million floating-point operations per second)
  - o C1 XL - 4 MIPS (million instructions per second)
  - o C1 XP - 6 MIPS

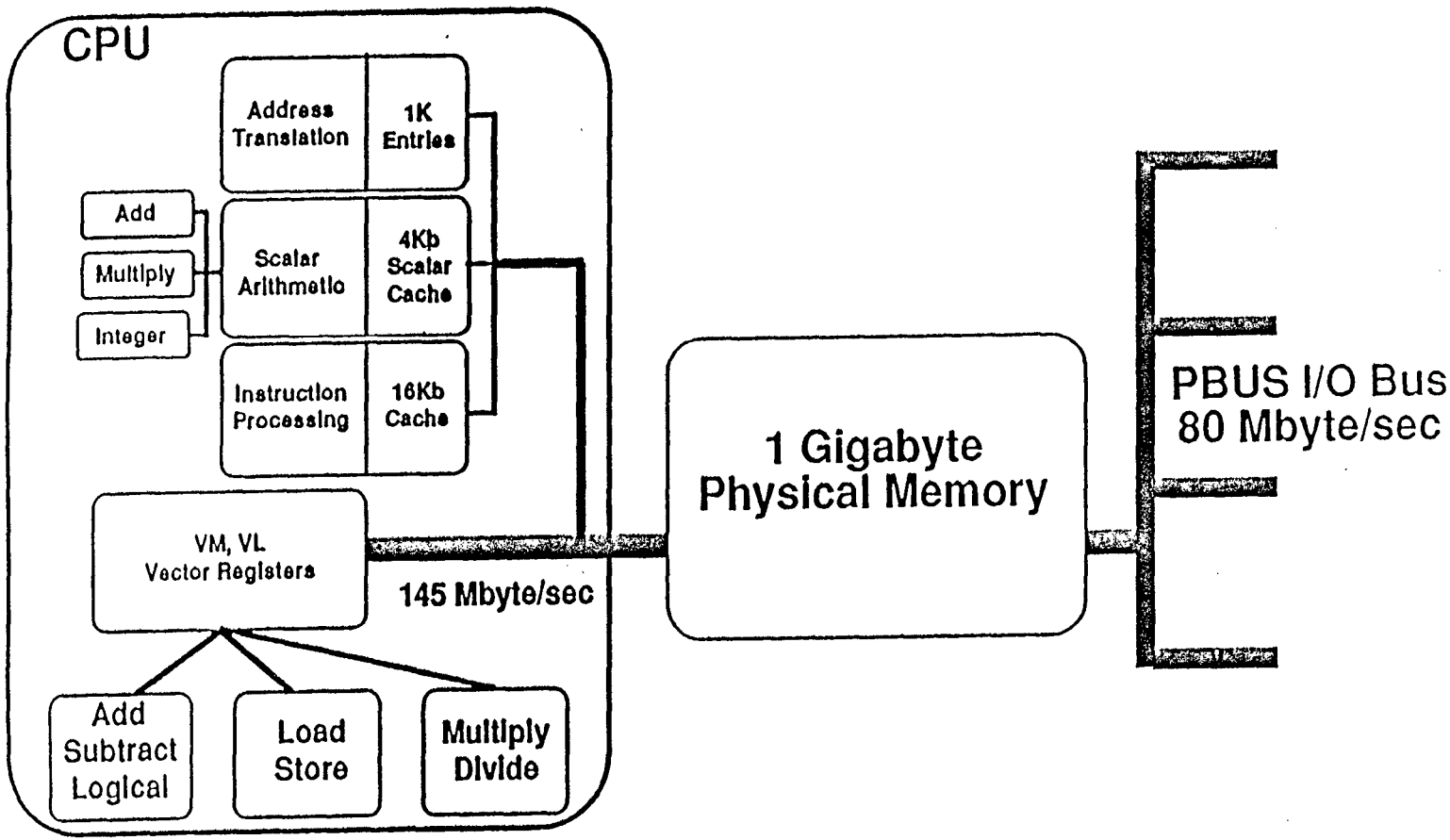
# C1 I/O SYSTEM

- System may include from one to five 68000-based channel control units (CCU)
- Up to two MULTIBUS chassis per IOP (CCU)
- MULTIBUS bandwidth - 4 megabytes per second
- I/O bus (PBUS) bandwidth - 80 megabytes per second
- 68000-based service processor (SPU)
- Power - 3200 watts (standard); 4500 maximum
- AC power: 208VAC, 60 HZ, 3 phase, 15 amps

# C120 System



# C130 Processor



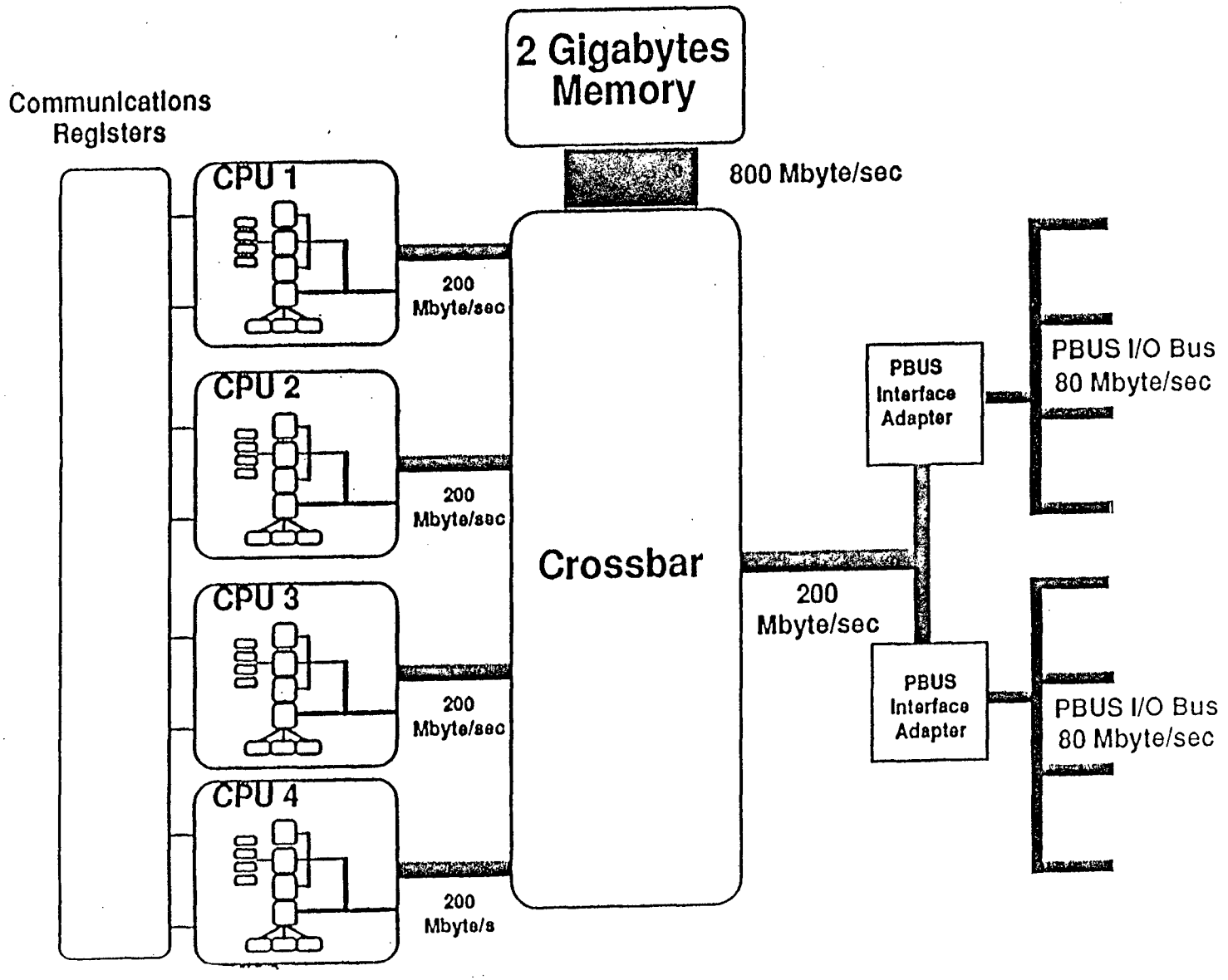
# C2 DESCRIPTION

- Processor is composed of the following 6 boards
  - o VPD Vector Processor Data
  - o VPC Vector Processor Control
  - o DCU Data Cache Unit
  - o SFU Scalar Functional Unit
  - o IPP Instruction Pre-Processor
  - o ASP Address Scalar Processor
- Service Processor (SP2)
  - o Second generation service processor
  - o Controls the system clocks and gates the clocks to the system boards
  - o Has the highest priority to the MCMs(memory units)
- System Utility Board (CPX)
  - o Contains the referenced and modified bits
  - o Contains the communication registers for the synchronization of processors (128 64-bit registers with locks)

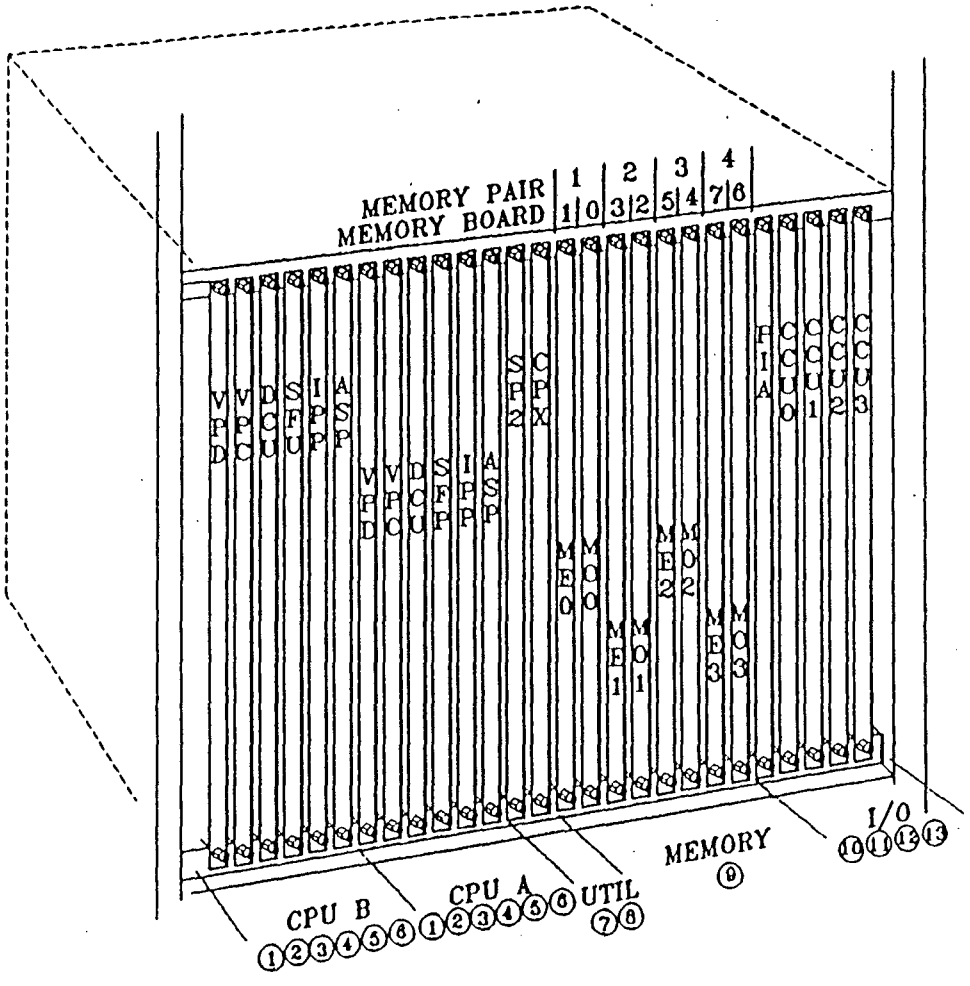
# C2 DESCRIPTION (CONT.)

- Memory system
  - o Composed of pairs of MCM boards (odd and even)
  - o Each MCM supports 32-bit data and parity
  - o Each MCM has 5 ports which support 200 Mbytes/second transfers
  - o All vector load/stores use the scalar processor to access memory
- Peripheral Interface Adaptor (PIA)
  - o Generates all the system clocks (40 ns)
  - o Interface to the E port from the memory system
  - o Provides arbitration for SP2 and I/O (SP2 > I/O)
- Channel Control Units (CCU)
  - o System supports 4 boards of the following types: HSP, IDS, IOP, and VIOP

# C2XX Processors

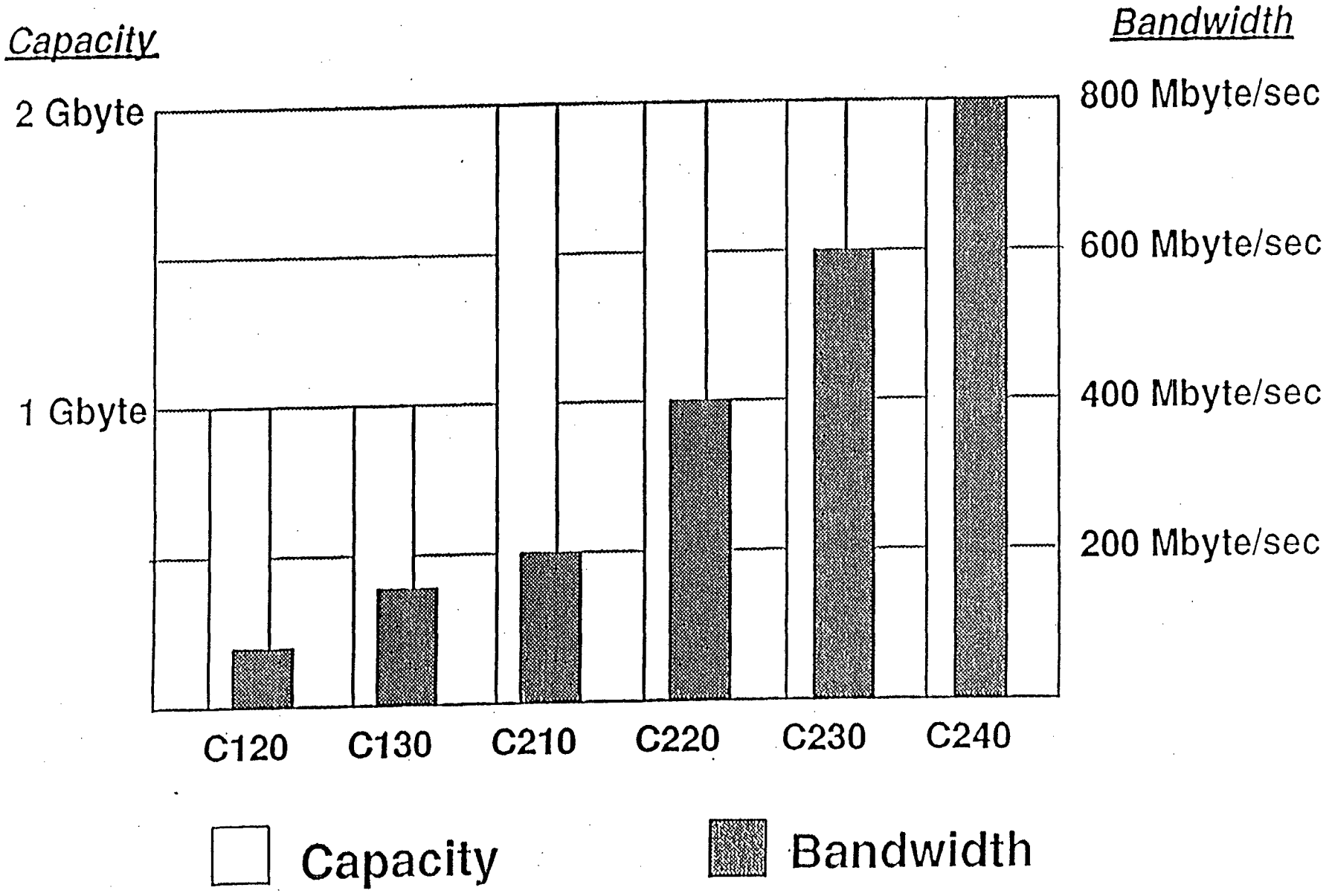


# C201,C202,C210,C220 CARD CAGE




1 VPD – VECTOR PROC. DATA
2 VPC – VECTOR PROC. CONTROL
3 DCU – DATA CACHE UNIT
4 SFU – SCALAR FUNCTION UNIT
5 IPP – INST. PROC. UNIT
6 ASP – ADDRESS SCALAR PROC.
7 SP2 – SERVICE PROCESSOR 2
8 CPX – CPU UTILITY
9 MCM – MEMORY UNIT
10 PIA – PBUS INTERFACE ADAPT
11 MIVP – MULTIBUS I/O PROCESSOR
12 VIVP – VMEBUS I/O PROCESSOR
13 HSP – HIGH SPEED PROCESSOR

# C Series Memory Capacity



# THE BOOTING PROCESS

- You can power-up or shutdown to any of the following operational environments:
  - o Front Panel
  - o Service Processor Unit
  - o ConvexOS single-user mode
  - o ConvexOS multi-user mode

# PROCESSOR OPERATIONAL ENVIRONMENT

- Power-up/power-down
  - o Power-up mode - normal mode
  - o Power-down mode
- Soft Front Panel (firmware)
  - o Modify settings of the firmware
  - o Select system boot options
- Service Processor Unit (SPU) UNIX
  - o Controls diagnostic software
  - o Coordinates error logging
  - o Boots ConvexOS
- ConvexOS single-user
  - o Superuser executes commands
  - o Only the root partition is mounted
- ConvexOS multi-user
  - o All partitions are mounted
  - o Users may login

# POWERING UP THE SYSTEM

- Proceedure for powering up from complete shutdown:
  - o On the CPU bay, turn the AC power controller mode switch to *REMOTE*
  - o Turn on the system breaker
  - o Do the same to the other cabinets
  - o Turn on the tapes, first the breaker, then the power switch in front
  - o Turn on the disks, first the breaker, then the power switch in front
  - o Turn on console
  - o Select the appropriate keyswitch setting

# FRONT PANEL OPERATION

- The Front Panel is located at the top of the leftmost bay, inside the front door
- The Front Panel contains a keyswitch, a reset switch, and several indicators:
  - o Attention
  - o Power On
  - o Run
- The Keyswitch has 4 positions:
  - o Off
  - o Local Maintenance
  - o Secure
  - o Remote Maintenance
- The reset switch is used to force the system to reboot

# KEYSWITCH SETTINGS

The keyswitch has four positions:

- *OFF*
- *LOCAL MAINTENANCE*
  - o System boots the Soft Front Panel on power-up
  - o System reset button is enabled
  - o Access to the SPU (with ^P) is allowed at system console
- *REMOTE MAINTENANCE*
  - o Similar to *LOCAL MAINTENANCE*
  - o Commands go through the SPU modem port
- *SECURE EXECUTION*
  - o Boots to multi-user mode when automatic boot is enabled at the Soft Front Panel
  - o System reset button is disabled
  - o Access to the SPU is disabled
  - o System console is disabled

# BOOTING THE SOFT FRONT PANEL

- The Soft Front Panel is used to control firmware operations
- To boot the Soft Front Panel (after power-down), turn the keyswitch from *OFF* to *LOCAL MAINTENANCE*
- The default Soft Front Panel display:

```
Convex-1 Front Panel / Module Rev: 2.0, Version: 1 / CPU SN 17
mode-of-operation = normal-os          boot-device = disk
location-of-bootstrap = default        power-up-reboot = enable
automatic-reboot = enable              spu-selftest = disable
os-flags = 0                           remote-port-bps = 1200
(fp)>
```

- The Soft Front Panel prompt: `(fp)>`
  - o The erase character: <DEL>
- Alternate ways to display the front panel:
  - o Set Soft Front Panel option *power-up-reboot* to `disable` (stops at Front Panel on power interruption)
  - o Press reset button when keyswitch is in LOCAL MAINTENANCE
  - o File inconsisteny errors will be caused on ConvexOS and SPU UNIX by pressing the reset button while ConvexOS is running

# SOFT FRONT PANEL COMMANDS

- Basic commands:

| | |
|---|---|
| `b or boot` | Boots SPU and ConvexOS |
| `d or display` | Displays switch settings |
| `h or help` | Displays a help screen |
| `s or set` | Sets panel switches |

- Format of the *s* or *set* command:

  (fp)> s *switch* = *value*

- The *switch* and *value* can be abbreviated by using the first character of *switch* and the first character of *value*

```
(fp)> set mode = normal-os
        or
(fp)> s m = n
```

# SOFT FRONT PANEL SWITCHES

- *b* or *boot-device* settings are:

| | |
|---|---|
| d or disk | Boot from disk |
| t or tape | Boot from tape |

- *l* or *location-of-boot-strap* settings are:

| | |
|---|---|
| d or default | Location of bootstrap |
| 1,2,3 | Three backup locations |

- *p* or *power-up-reboot* settings are:

| | |
|---|---|
| d or disable | No automatic reboot |
| e or enable | Forces automatic reboot |

- *a* or *automatic-reboot* settings are:

| | |
|---|---|
| d or disable | Booting disabled at Front Panel |
| e or enable | Booting enabled at Front Panel |

- *m* or *mode-of-operation* settings are:

| | |
|---|---|
| n or normal-os | Boot multi-user |
| a or alternate-os | Menu selection |
| d or diagnostic | Run diagnostics |

# BOOTING SPU UNIX

- SPU UNIX starts up from the Soft Front Panel
- The "automatic-reboot" flag starts SPU UNIX automatically when the Keyswitch is in the "secure" position
- The "boot" command starts SPU UNIX at the Soft Front Panel
- SPU UNIX boots from the 20MB SPU Winchester disk
- (spu)> is the prompt for SPU UNIX
- UNIX V7 Bourne Shell is the command interpreter for the SPU

# BOOTING ConvexOS

- Programs to boot ConvexOS from SPU UNIX are kept in */mnt/os* on the SPU UNIX file system
- A shell program */mnt/os/bootcmd* executes the boot programs
- To boot ConvexOS to multi-user mode enter the following SPU UNIX commands:

```
(spu)> cd /mnt/os
(spu)> boot
```

- The file system check program *fsck* is run when booting to multi-user mode from the (spu)>
- To boot ConvexOS to single user mode enter the following SPU UNIX commands:

```
(spu)> cd /mnt/os
(spu)> boot single
```

- *fsck* does not run when booting to single-user mode
- *fsck* may be run manually (if appropriate) before going to multi-user mode

# ConvexOS BOOT PROCESS TO MULTI-USER

- The hardware is initialized
- The operating system is loaded from the SPU UNIX file */mnt/os/vmunix*
- ConvexOS uses the SPU file */ioconfig* to determine the peripheral configuration
- IOP software is loaded in the I/O processors from the SPU file */mnt/os/iop*
- The */mnt/errlog* is started
- The ConvexOS root file system is checked and mounted
- The ConvexOS */etc/init* program is started
- Various daemons and accounting are started from the */etc/rc*, */etc/rc.local* and */etc/rc.std* files
- The multi-user (login) prompt is displayed

# SYSTEM SHUTDOWN

- Use the */etc/shutdown* program to terminate ConvexOS

- *shutdown* command format:

  `/etc/shutdown` [*options*] *time* [*warning_message*]

- *shutdown* options:

  | | |
  |---|---|
  | `-k` | Don't shut the system down. Only print the warning messages |
  | `-h` | Halt the CPU (shutdown to SPU) |
  | `-r` | Shutdown with automatic reboot |

- The shutdown *time* can be passed as a parameter in one of three formats:

  | | |
  |---|---|
  | +*minutes* | Bring it down in *minutes* minutes |
  | *hour*:*min* | Bring it down at a specific time |
  | `now` | Bring it down immediately |

- To shutdown to single-user mode:
  `/etc/shutdown now "For system upgrade"`

- To shutdown to the SPU:
  `/etc/shutdown -h +15 "For hardware test"`

# SYSTEM SHUTDOWN (Continued)

- *shutdown* will use *wall* to periodically warn users of the impending shutdown
- 5 minutes before shutdown the file */etc/nologin* is created containing the shutdown message
  - o Only superuser can login
  - o Normal users will be denied permission to login
- Process ID for *shutdown* is displayed when *shutdown* is invoked
- *shutdown* can be terminated with the *kill* command
- Before completing *shutdown* places a record containing the date and the reason for the shutdown in the file */usr/adm/shutdownlog*

# SAMPLE SHUTDOWN

```
# shutdown -h now 'going down for pm'

Shutdown at 09:56 (in 0 minutes) [pid 115]
#
*** FINAL System shutdown message
   from root@Customer ***
System going down IMMEDIATELY
    ... going down for pm
System shutdown time has arrived
[_cpu@09:56:20] syncing disks
[_cpu@09:56:20] done
[_cpu@09:56:21] halting in tight loop:
/mnt/os/boot_cpu: 47 Terminated

Cleaning up SPU processes...
boot: 50 Killed
(spu)>
```

# OTHER SHUTDOWN COMMANDS

- There are two other commands associated with shutdown */etc/halt* and */etc/reboot*
  - o *halt* stops the processor and brings the system down to the SPU
  - o *reboot* causes the system to shutdown and reboot ConvexOS
- These commands can be given the options
  - o *-n* which causes the system not to sync the disks
  - o *-q* does a quick ungraceful shutdown where running processes are not terminated

# SHUTDOWN TO THE FRONT PANEL

- Turn the keyswitch to LOCAL position
- Shutdown to the SPU:

  # shutdown -h now

- At the (spu)> prompt enter the command */etc/reboot*

  (spu)> /etc/reboot

  The (fp)> prompt is displayed

# POWER DOWN FROM ConvexOS

- Enter the shutdown command to halt the CPU from multi-user mode:

```
# /etc/shutdown -h +10 "Hardware
 upgrade"
```

- At the console (spu)> prompt, enter the power down command (*pwrdwn*):

```
(spu)> pwrdwn
```

- When the *Ready for powerdown message* is displayed:
  - o Turn off appropriate devices
  - o Set keyswitch to OFF
  - o Move AC breaker to off (if necessary)

- Put keyswitch in LOCAL position to display SOFT FRONT PANEL on power up

# SHUTDOWN/REBOOT WITH SYSTEM HANG

- If the system hangs and it is not possible to execute a *shutdown* command
  - o Set the system console keyswitch to LOCAL
  - o Access the SPU by entering CTRL P
- Kill SPU processes gracefully:

```
(spu)> osclean
(spu)> sysreset
```

- Reboot with the command:

```
(spu)> boot
```

# ACCESSING THE SPU FROM ConvexOS

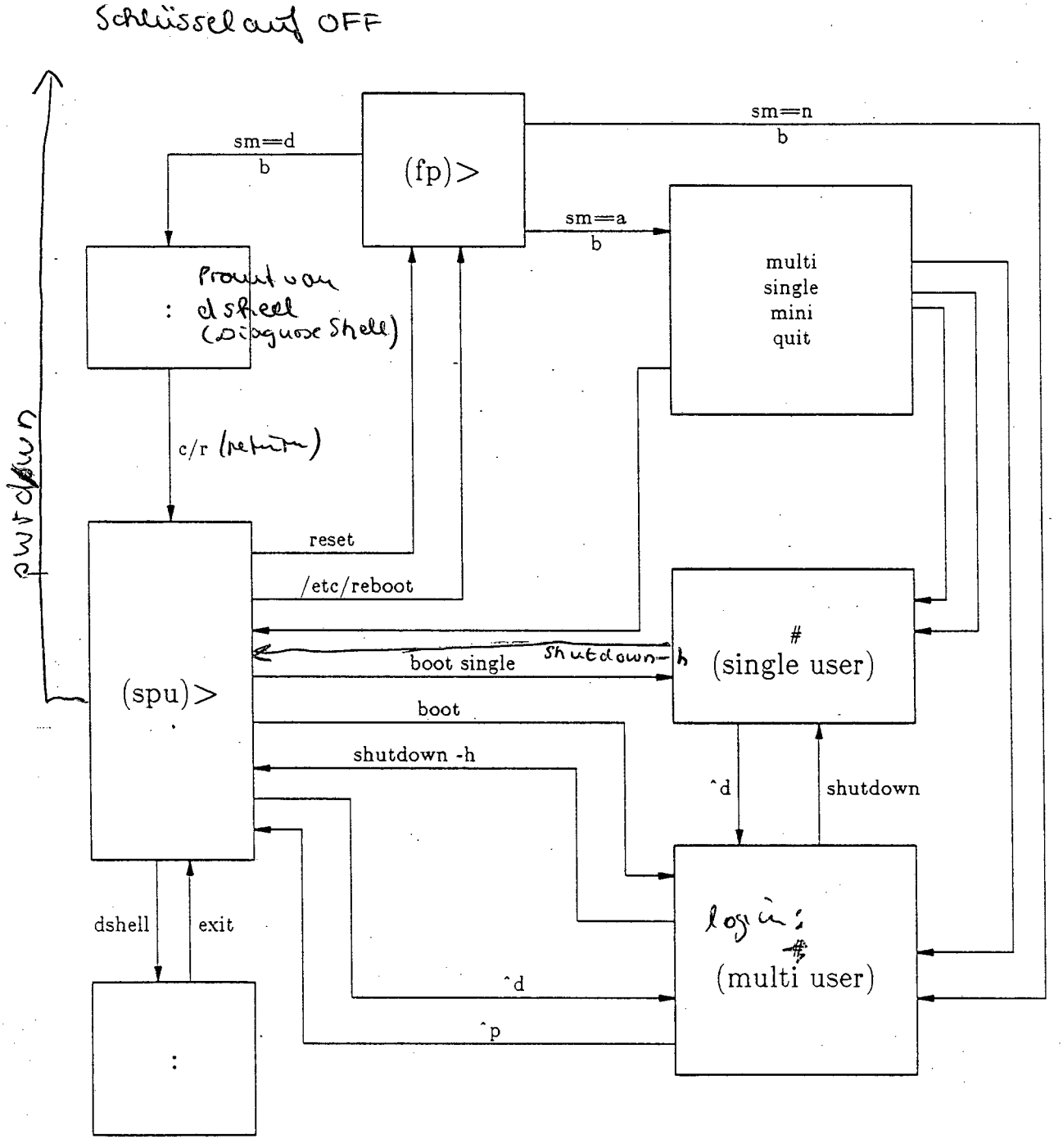
- To access the SPU while ConvexOS is running, enter CTRL P (^P) at the system console

- To return to multi-user mode (while ConvexOS is running), enter CTRL D (^D) at the system console

***WARNING***

- Some SPU commands can cause the system to crash if executed while ConvexOS is running

# SYSTEM PROMPTS



# BOOT PROGRAMS & FILES

## Programs and files used in the boot process

| Program or File | Location | Function |
|---|---|---|
| bootcmd | SPU | Selects mode of boot |
| bootcmd.local | SPU | Modifies boot parameters |
| /etc/init | ConvexOS | First process in ConvexOS<br>Begins all other processes |
| /etc/rc | ConvexOS | Sets up OS operation<br>Starts accounting<br>Calls rc.local |
| /etc/rc.local | ConvexOS | Set up local OS requirements |
| /etc/rc.std | ConvexOS | Set up local daemons |
| /etc/fsck | SPU & ConvexOS | Checks the disk file systems at boot time |
| /ioconfig | SPU | identifies I/O devices attached to the system |
| /mnt/errlog | SPU | records boot operations and system errors |

# THE */etc/init* PROGRAM

- The */etc/init* program is run by the operation system as process number *1*
- All processes except system kernel processes are descended from *init*
- *init* does the following:
  - o Executes the commands listed in */etc/.initrc*
  - o If the *boot single* command had been requested, *init* performs the following operations:
    - * Starts superuser shell on console
    - * Interfaces with the superuser
    - * Continues at shell termination
  - o Creates a process for each terminal port for login
  - o Executes commands from the file */etc/rc* and */etc/rc.local* files
  - o *init* calls *getty* which enables login ports
- The multi-user prompt (login) appears on the console

# /etc/rc (page 1)

**The /etc/rc file is a Bourne shell script used to initialize the operating system.**

```
# redirect all output to the console
exec > /dev/console 2>&1
```

*The first line is a comment line in the Bourne shell, the second line redirects the output of the command lines to the console.*

```
HOME=/ ; export HOME
```

*This line sets the variable "HOME" as /, then passes the variable to the interpreter.*

```
PATH=/bin:/usr/bin
```

*This line sets the "PATH" variable as /bin and /usr/bin.*

```
#
# Reset timezone
```

*Comment lines*

```
if [ -r /etc/timezone_name ]
then
     date -z `cat /etc/timezone_name`
fi
```

*This if then block tests to see if the file "/etc/timezone_name" is readable. [ -r /etc/timezone_name ] If the file is readable, the contents are used with the date command to set the timezone to that specified in the file. The back quotes cause the result of the cat command to be passed as an option to the date command. The fi is the end of an if statement in the Bourne shell.*

```
#
# load stripe filesystem tables
```

# /etc/rc (page 2)

```
if [ -r /etc/stripecap ]
then
     /etc/putst -a
fi
```

*This if then block tests to see if striped files systems are defined in the /etc/stripecap file. ( stripecap is built when the newst command is run. ) If the file is readable, the putst command passes these definitions to the kernel.*

```
#
```

*The next section provides for variations in the handling of the file systems during the boot porcess.*

```
if [ -r /fastboot ]
then
     rm -f /fastboot
     echo 'Fast boot... skipping disk checks'
```

*If the file /fastboot is readable, (it is created with the fasthalt command ), the file is removed. The echo message appears on the console.*

```
elif [ $1x = autobootx ]
then
     echo 'Automatic reboot in progress...'
     date
```

*The echo message appears on the screen and the date command is executed.*

```
     /etc/preen
```

*The /etc/preen program is run. This program does a file system consistency check on all the 4.2 type file systems found in the fstab file.*

# /etc/rc (page 3)

```
case $? in
0)
     date
     ;;
4)
     /etc/reboot -n
     ;;
8)
     echo 'Automatic reboot failed... help!'
     exit 1
     ;;
12)
     echo 'Reboot interrupted'
     exit 1
     ;;
*)
     echo 'Unknown error in reboot'
     exit 1
     ;;
esac
```

*The exit status of the /etc/preen program is checked. A normal exit would have a value of 0, after which the date commad is executed. A status of 4 would indicate the root filesystem had an error discovered during the filesystem check and the boot process is started over again. A status of 8 indicates an error occured during the file system checks of another file in fstab that must be corrected by the operator attempting to boot the system. A status of 12 indicates the operator interrupted the boot process. This is usually done with a control c at the console. Any other status, indicated by the *, is undefined and assumed to be an error. The "exit 1" lines are used to exit the rc script with a value of 1. This would return to single user at this point.*

```
else
     date
```

*If date has not been run from another case, then execute the date command.*

# /etc/rc (page 4)

```
fi
```

*End of the if then block.*

```
rm -f /etc/nologin
```

*The nologin file is created with the shutdown command to prevent users from logging in while the system is preparing to shutdown. Even if the file exist, root can login before the system shuts down. Root could kill the process id of the shutdown command. This file needs to be removed before users can login into the system.*

```
# attempt to rationally recover the passwd file if needed
```

*The /etc/ptmp file is created when modifications are being made to the password file with the password command, the nu command, or the vipw command. In the event the system crashed while one of these commands is in use then the file would still exist. It is used as a backup copy and as a locking mechanism for these commands. The status of the ptmp and the passwd files are compared, and if necessary, ptmp is used to rebuild the passwd file. The -s test is used to test if the file size is greater than 0.*

```
if [ -s /etc/ptmp ]
then
     if [ -s /etc/passwd ]
     then
          ls -l /etc/passwd /etc/ptmp
          rm -f /etc/ptmp      # should really remove the shorter
     else
          echo 'passwd file recovered from ptmp'
          mv /etc/ptmp /etc/passwd
     fi
elif [ -r /etc/ptmp ]
then
     echo 'removing passwd lock file'
     rm -f /etc/ptmp
fi
```

# /etc/rc (page 5)

```
/etc/umount -a > /dev/null 2>&1
```

*All files systems are unmounted. In the event of an unusual shutdown, this step closes remotely mounted filesystems. It serves to notify other hosts on the network to close the ports that serve the network filesysems to this host.*

```
cp /dev/null /etc/mtab
```

*This clears the mount table.*

```
/etc/swapon -a
```

*This adds all swap areas defined in fstab.*

```
/etc/mount -f /
```

*This forces the mount of the root partition.*

```
/etc/mount -at 4.2
```

*This attaches all 4.2 type filesystems.*

```
# attempt to recover 8.0 nqsdaemon and qmgr

if [ -f /usr/lib/nqs/nqsdaemon ]; then
  if cmp /usr/lib/nqs/nqsdaemon /usr/lib/nqs/nqsdaemon.8.0
  > /dev/null 2>&1

  then
    echo
  else
    echo 're-installing nqsdaemon'
    cp /usr/lib/nqs/nqsdaemon.8.0 /usr/lib/nqs/nqsdaemon
  fi
fi
```

*This if then block checks the status of nqsdeamon. It is compared to the 8.0 version to insure integrity, and replaced if corrupted.*

```
if [ -f /usr/convex/gmgr ]; then
  if cmp /usr/convex/qmgr /usr/convex/qmgr.8.0 > /dev/null 2>&1
```

# /etc/rc (page 6)

```
  then
    echo
  else
    echo 're-installing qmgr'
    cp /usr/convex/qmgr.8.0 /usr/convex/qmgr
  fi
fi
```

*This if then block checks the status of qmgr. It is compared to the 8.0 version to insure integrity, and replaced if corrupted.*

```
if [ -f /etc/rc.local ]; then
     sh /etc/rc.local
else
     echo 'Cannot find /etc/rc.local'
fi
```

*This if then block tests to see if /etc/rc.local is a file. If it is, a shell is created to run the contents of rc.local. The rc program continues after rc.local has finished. If there is no rc.local, notify the console that no rc.local is available.*

```
/etc/ifconfig lo0 localhost up
```

*Ifconfig is used to bring up the loopback network. This is used to support talk and other utilities, even if a network is not installed.*

```
cd /tmp
echo 'preserving editor files'
/usr/lib/ex3.7preserve -a
```

*The working directory is changed to /tmp, where the preserve program looks for the image buffers used with editors. If buffers are found, the user is notified by mail, with instructions for the recovery of the most recent changes to a file which was being edited when the system halted. ( used for crash recovery)*

# /etc/rc (page 7)

```
if [ -f /etc/rc.std ]; then
     sh /etc/rc.std
else
     echo 'Cannot find /etc/rc.std'
fi
```

*Test to see if rc.std is a file and run the contents in a Bourne shell if the file exist. Otherwise notify the console that there is no rc.std to run.*

```
echo 'initializing accounting'
/etc/accton /usr/adm/acct
```

*Turn on the accounting system and use the /usr/adm/acct directory to hold the information collected by the accounting program.*

```
cd /
date
exit 0
```

*Change the working directory to root, execute the date command, then exit rc with a status of 0.*

# /etc/rc.local (page 1)

The /etc/rc.local file is a Bourne shell script used to initialize the operating system. This script is called from rc and is used to reflect specific changes for a system.

```
PATH=/etc:/usr/etc:/bin:/usr/ucb:/usr/bin:/usr/convex
export PATH
```

*The search path is expanded.*

```
/bin/hostname convext
```

*This line sets the hostname. In this example the system name is convext. The system administrator edits this file to enter the hostname of the system.*

```
if [ -f /bin/domainname ]; then
     /bin/domainname ""          # only set if Yellow Pages is used
fi
```

*This if then block tests to see if the /bin/domain file exists. If yellow pages is in use, your domain name needs to be added between the quotes.*

```
if [ -f /etc/ifconfig ]; then
     /etc/ifconfig ex0 `/bin/hostname` up netmask 0xffffff00 \
             >/dev/console
     # All physical interfaces MUST be configured before lo0 is configu
     /etc/ifconfig lo0 localhost up              >/dev/console
fi
```

*ifconfig is used to initialize the network, ex0 refers to a hardware controller. Each controller must be initalized at boot time, additional entries may be needed for additional controllers.*

# /etc/rc.local (page 2)

```
# syslogd must be started before any other daemons.

if [ -f /usr/etc/syslogd ]; then
     rm -f /dev/log
     /usr/etc/syslogd & echo 'starting system logger'  >/dev/console
fi
```

*The syslog daemon places entries into the log file as specified in /etc/syslog.conf.*

```
echo -n 'starting rpc and net services:'          >/dev/console
if [ -f /etc/portmap ]; then
     /etc/portmap & echo -n ' portmap'          >/dev/console
fi
```

*The portmap daemon is used to assign Remote Procedure Calls (RPS) to a port number in the server.*

```
if [ -f /bin/domainname -a "`/bin/domainname`" != "" ]; then
     if [ -f /usr/etc/ypserv -a -d /etc/yp/`/bin/domainname` ]; then
          /usr/etc/ypserv & echo -n ' ypserv'    >/dev/console
     fi
     if [ -f /etc/ypbind ]; then
          /etc/ypbind & echo -n ' ypbind'      >/dev/console
     fi
fi
```

*If yellow pages are in use and the domain name is set, the yellow page daemons are started.*

```
if [ -f /etc/routed ]; then
     /etc/routed & echo -n ' routed'          >/dev/console
fi
```

*The route daemon is used to manage the address route tables for Network File Systems (NFS).*

```
if [ -f /etc/biod ]; then
     /etc/biod 4 & echo -n ' biod'            >/dev/console
fi
```

# /etc/rc.local (page 3)

*biod is the block i/o daemon that handles client request in NFS.*

```
if [ -f /etc/nfsd -a -f /etc/exports -a -f /usr/etc/exportfs ]; then
     /etc/nfsd 4 & echo -n ' nfsd'            >/dev/console
     >/etc/xtab; /usr/etc/exportfs -a &
fi
```

*nfsd is the NFS daemon used on the server to handle NFS requests.*

```
echo '.'                                  >/dev/console

/etc/umount -at nfs
```

*All nfs type file systems are unmounted. If the system goes down without closing the RPCs, this will notify the servers to close any open RPCs.*

```
/etc/mount -vat nfs &                     >/dev/console 2>&1
```

*After all RPCs are closed on the servers, the network file systems are attached.*

```
echo -n 'checking quotas: '                    >/dev/console
     /usr/etc/quotacheck -p -a           >/dev/console 2>&1
     /usr/etc/quotaon -a
echo 'done.'                              >/dev/console
```

*If quota file systems are set up, quotas are turned on.*

```
echo -n 'local daemons:'                       >/dev/console
if [ -f /etc/rwhod ]; then
     /etc/rwhod & echo -n ' rwhod'             >/dev/console
fi
```

*The rwho daemon is started. This daemon maintains the databases used by the rwho and ruptime commands.*

```
if [ -f /etc/stat ]; then
     /etc/stat & echo -n ' statistics'         >/dev/console
fi
```

# /etc/rc.local (page 4)

*stat is used to obtain information about files in the system.*

```
if [ -f /usr/lib/nqs/nqsdaemon ]; then
  /usr/lib/nqs/nqsdaemon & echo -n ' CXbatch'      >/dev/console
fi
```

*The nqsdaemon handles local CXbatch activities such as submit, or list.*

```
if [ -f /etc/rpc.statd -a -f /etc/rpc.lockd ]; then
     /etc/rpc.statd & echo -n ' statd'          >/dev/console
     /etc/rpc.lockd & echo -n ' lockd'          >/dev/console
fi
```

*statd and lockd provide crash recovery for the NFS services.*

```
if [ -d /tftpboot ]; then
     if [ -f /etc/ethers -a -f /etc/arp ]; then
          /etc/arp -e /etc/ethers & echo -n ' rarp' >/dev/console
     fi
     if [ -f /usr/etc/ndbootd -a -f /tftpboot/sun2.bb ]; then
          /usr/etc/ndbootd & echo -n ' ndbootd'  >/dev/console
     fi
fi
```

*If the software from Sun Microsystems for the support of diskless workstations is on the system, this allows connection to Convex by the workstations. NOTE: this software is not distributed by Convex, but may be obtained from Sun Microsystems.*

```
echo '.'                                   >/dev/console
```

# /etc/rc.std (page 1)

The /etc/rc.std file is a Bourne shell script used to initialize the operating system. This script is called from rc and is used to start standard daemons.

```
# redirect all output to the console
exec > /dev/console 2>&1
PATH=/etc:/usr/etc:/bin:/usr/ucb:/usr/bin:/usr/convex
export PATH

echo -n 'standard daemons:'

/etc/update; echo -n ' update'
```

*The update command is used to periodically update the superblocks on the system disk drives.*

```
/etc/cron; echo -n ' cron'
```

*The cron utility builds a table in the kernel, which executes the items found in crontab. This utility looks for .crontab files in users home directories for additional items to be done.*

```
if [ -f /etc/inetd ]; then
     /etc/inetd & echo -n ' inetd'
fi
```

*The inet daemon looks at /etc/inetd.conf (by default) for a list of services to be supported. This allows one daemon to run that can invoke programs on request, thus reducing the system load.*

```
if [ -f /usr/lib/sendmail ]; then
     (cd /usr/spool/mqueue; rm -f lf*)
     /usr/lib/sendmail -bd -q1h & echo -n ' sendmail'
fi
```

*Sendmail is the utility that delivers preformated messages to the appropriate receiver.*

# /etc/rc.std (page 2)

```
cd /usr/spool; rm -f uucp/LCK/* uucp/TM./*
```

*The LCK and TM directories contain files used to manage uucp connections. Files are created to prevent the use of a connection by multiple programs or users.*

```
if [ -f /usr/lib/lpd ]; then
     rm -f /dev/printer
     /usr/lib/lpd & echo -n ' printer'
fi
```

*The lpd is the line printer daemon that manages spooling and printing of jobs to the system printers.*

```
if [ -f /usr/lib/opreq/opreq_daemon ]; then
     /usr/lib/opreq/opreq_daemon & echo -n ' opreq_daemon'
fi
```

*The opreq_daemon is the main utility that supports the opreq system usage. This includes such items as mount requests.*

```
if [ -f /usr/lib/tape/tpdaemon ]; then
     /usr/lib/tape/tpdaemon & echo -n ' tpdaemon'
fi
```

*The tpdaemon is the main tape system daemon.*

```
echo '.'
```

# THE SPU */ioconfig* FILE

- */ioconfig* is used by the boot program to determine hardware to check for
- The file contains four types of records:
  - o IOP/VIOP:

```
iop/viop number
iop/viop 7
```

  - o Multibus/VMEbus:

```
mbus/vme number
mbus/vme 0
```

  - o I/O controller:

```
ctlr type csr address int level
ctlr DKC-001 csr 0x3f0 int 2
ctlr DKC-203 csr 0x200 int 1
```

  - o I/O unit:

```
unit number type name
unit 0 type DKD-001
unit 0 type DKD-208
```

- See *System Manager's Guide* for list of controller and device specifications

## */ioconfig* (Continued)

- Example section of */ioconfig*:

```
iop 7
 mbus 0
  ctlr DKC-001 csr 0x3f0 int 2
   unit 0 type DKD-001
  ctlr MTC-001 csr 0x0c0 int 4
   unit 0 type MTD-001
 mbus 1
  ctlr DKC-001 csr 0x3f0 int 2
   unit 0 type DKD-001
  ctlr MTC-001 csr 0x0c0 int 4
   unit 0 type MTD-001
  ctlr LAN-001 csr 0x4c0 int 1
   unit 0 type ex
viop 6
 vme 0
  ctlr DKC-204 csr 0x200 int 1
   unit 0 type DKD-206
   unit 1 type DKD-206
  ctlr DKC-203 at csr 0x000 int 1
   unit 0 type DKD-208
   unit 1 type DKD-208
  ctlr PRC-001 csr 0x2c0 int 0
   unit 0 type PRT-001
  ctlr PRC-001 csr 0x4c0 int 1
   unit 1 type PRT-001
```

# THE SPU */mnt/errlog* FILE

- This file contains all the hard and soft errors reported by the CPU, the SPU, and the I/O processors
- It should periodically be copied to the ConvexOS file system for inspection and archival

```
# spu -r /mnt/errlog > errlog.hold
```

- It should also be truncated periodically

```
# spu -w /mnt/errlog < /dev/null
# spucmd cp /dev/null /mnt/errlog
```

# NU UTILITY

- The *nu* utility automates adding new users
- This utility automatically updates the following files when a user is added:
  - o */etc/passwd*
  - o */etc/pwrestrict*
  - o */etc/uidcount*
- *nu* builds each user's home directory and provides the user with the following files:

  *.cshrc*
  *.exrc*
  *.login*
  *.logout*
  *.project*

- These files are located in the */usr/skel* directory
- Modify these files or create additional files to customize the user environment

# THE *nurc* FILE

- Default values are provided by the *nu* utility for:
  - o *user id, group, home directory, home directory protection, login shell*
- To change the default values, create the */etc/nurc* file
- Each entry in *nurc* defines a default value in the format:

  *fieldname*: *value*

- Some *fieldname* parameters that can be modified:

| | |
|---|---|
| `directory` | Path to home directory |
| `protection` | Home directory protection |
| `group` | Group id |
| `shell` | Login shell |
| `username` | Full name of user |
| `minwks` | Minimum number of weeks for password aging |
| `office` | Location of office |
| `extension` | Office phone extension number |
| `maxwks` | Maximum number of weeks for password aging |
| `typed` | Enforce restrictions |

# THE *nurc* FILE (Continued)

| | |
|---|---|
| `quota` | Set up quota restrictions on home directory |
| `diskquota` | Soft and hard restrictions for file and inodes |
| `skeleton` | Directory containing files that copied to each user's home directory |

- *value* is the default value for the parameter
- Example *nurc* file:

```
directory: /tac
protection: 0700
group: 49
typed
minwks: 2
maxwks: 4
```

# USING *nu*

- **The *nu* utility adds users interactively:**

```
# nu
Login name: rocky
User id [800]:
Group id [staff]:
Home directory [/mnt/rocky]:
Home directory protection [0755]: 700
Login shell [/bin/csh]:

Changing the user information...
Default values are printed inside of of '[]'.
To accept the default, type <return>.
To have a blank entry, type the word 'none'.

Name [username]: Rocky Raccoon
Room number (Exs: 18A or 17B) [office]: 546
Office Phone (Ex: 223) [extension]: 798
Home Phone (Ex: 6610379) [homephone]: none

Password to be typed [N]:
Password subject to aging [N]: Y
  Enter the min period for the password[1]:2
  Enter the max period for the password[52]:4

Entering the user password...
New password:
Retype new password:

Creating the home directory...
Successful completion
#
```

# BATCH FILE FOR *nu*

- To add several new users using the *nu* utility with no interaction, create a batch input file
- The batch file defines user-specific fields
- The fields that can be defined include those for the *nurc* file and one additional field--*login*
- Fields that are common to a large group of new users are placed in the *nurc* file
- The necessary user-specific fields are placed in the batchfile
- The format for the batch file:

  *fieldname=value*

Example batch file entry:

```
login=jwild:username=John Wild:office=100:extension=235
login=jdopey:username=James Dopey:office=250:extension=895
```

- To invoke *nu* with a batch file:

  # nu -f *batch_filename*

# ADDING USERS MANUALLY

- Create a */etc/passwd* entry with the */etc/vipw* utility:

  `jones::250:100:John Jones24,456,9789090:`*/mnt/jones*`:/bin/csh`

- Leave the password file blank and add the password using the *passwd* utility

  ```
  #passwd jones
  New Password:
  Retype new password:
  ```

- If an undefined group has been assigned to the user, add the groupname and ID number to the */etc/group* file:

  `devmnt:*: 100:`

- Create the login directory

  `#mkdir` *home_dir_name*

- Create the login shell initialization files

  `#cp /usr/skel/.[a-z]*` *home_dir_name*

- Change ownership of the login directory and its contents to match the user's with the *chall* command

  `#chall -o`*username* `-g`*group* *home_dir_name*

- Add a password for the user with the *passwd* command

- Increment the uid count in the */etc/uidcount* file

# REMOVING USER ACCOUNTS

- Login for the user can be disabled by modifying the password field in the user's */etc/passwd* entry
- Change the 13 character encryption to '*'
- Mail aliases should be changed so the user's mail is forwarded to someone else
  /usr/lib/aliases
- The user's files can be archived and removed if they are not needed

# THE */etc/passwd* FILE

- */etc/passwd* contains login information for each user
- Users are added to the *passwd* file manually or by the *nu* utility
- Each entry contains seven colon-separated fields with the format:

  *name*:*password*:*UID*:*GID*:*comment*:*directory*:*program*

- *name* is the user's login name and is created by the system administrator
- *password* is encrypted password and is usually created initially by the system administrator
- *UID* is the user id number; the number is supplied by the */etc/uidcount* file
- *GID* is the group id number; the number is either the system default or one selected from the */etc/group* file

# THE */etc/passwd* FILE (Continued)

- *comment* is created initially by the system administrator; it can contain the following comma-separated subfields used by the *finger* utility:

  full name
  office number
  phone extension
  home phone number

- *directory* is the user's login directory

- *program* is the program that *login* is to exec after successful login (commonly */bin/csh*)

- Example *passwd* file entries:

```
                                  ← default group
root:ateeIZ3yncGlY:0:10:Superuser:/:/bin/csh
daemon:*:1:1:Our friend, the daemon:/:/bin/csh
resource:xx:3:10:Owns all resources:/:/bin/csh
anon:xx:4:40:Anon notes:/usr/spool/notes:/bin/csh
jones:OeAtIALrsykko:198:49:Joe Jones:/mnt/jones:
/bin/csh
```

# THE */etc/vipw* UTILITY

- */etc/vipw* is used to modify the *passwd* and *pwrestrict* file
- *vipw* uses the *vi* editor to edit *passwd*
- Another editor is used if specified in the EDITOR environment variable
  - o (`setenv EDITOR` *editor_pathname*)
- Only one user at a time can invoke *vipw*
- Temporary files are created (*/etc/ptmp*, */etc/rtmp*, and */etc/vtmp*) when *vipw* is invoked
- These temporary files should not exist outside */etc/vipw* unless a problem has occurred
- When a problem state exists such as an abnormal exit from *vipw* remove these files by using *rm*
- These temporary files are removed when the *passwd* file is written
- *vipw* tries to ensure that */etc/passwd* contains only correct entries

# PASSWORD AGING

- System password aging facility controls if and when a user changes passwords
- Password aging information is kept in the file */etc/pwrestrict*
- Entries are created using the *nu* (new user) utility
- Aging information is modified using the *vipw* utility
- When running the *nu* utility with password aging selected, the entry is automatically added to the */etc/pwrestrict* file
- The format of *pwrestrict* entries is:

  *name:password:age:typed:uid*

  *name* is the user's login name

  *password* is the encrypted password

# PASSWORD AGING (Continued)

- *age* contains 3 comma-separated subfields of aging information:

  *m*,*M*,*date*

  - o *m* is the minimum weeks between password changes
  - o *M* is the maximum weeks between password changes
  - o *m* and *M* = 0, the password must be changed at the next login
  - o If *m* is greater than *M*, only the superuser can change the password
  - o The date is supplied by the *nu* command

- The aging information is written in the *pwrestrict* file:

  *name*:*password*:2,4,Jan 27 1987:*type*:*uid*

- Use the *genrest* command to set up aging for all *passwd* entries (*pwrestrict* cannot exist):

```
genrest -m4 -M8
```

- Use *vipw* to modify existing or to add *pwrestrict* entries

# PASSWORD RESTRICTIONS

- *typed* determines if password restrictions are enforced
  - o Y enables this feature
  - o N disables this feature
  - o When restrictions are enabled, passwords must be at least 6 characters long
  - o Restrictions require two alphabetic characters and one numeric or special character
- *uid* is the user's UID
- A sample *pwrestrict* entry:

```
root:0eAtIALrsykko:2,4,Jan 27 1987:Y:0
```

# THE */etc/group* FILE

- */etc/group* determines to which groups a user belongs
  - o A user will belong to the default group specified in the user's */etc/passwd* entry
  - o */etc/group* can contain additional groups up to a maximum of 16 including the default
- Format:

```
group name:*:group ID:usernames
```

- Example:

```
operator:*:5:stadler,jones
guest:*:31:
vendor:*:48:bill,smith,hurtz
staff:*:49:anderson,johnson,bill,
          franks
```

# USER MAIL

- The system keeps a set of system-wide aliases in the file */usr/lib/aliases*
- Aliases are used by *sendmail* to create mail destinations
- Alias entries have the format:
  - o *alias*: *name1*, *name2*, ...
  - o *alias* must begin in column 1
  - o Lines beginning with white space are considered to be a continuation of the previous entry
- The name can be:
  - o Other aliases
  - o A program name specified in the format:

    " | *program*"
- A file name specified in the format:

  "> *filename*"

# USER MAIL (Continued)

- Users' mailboxes are located in the directory */usr/spool/mail*

- Rebuild the aliases database after any changes to */usr/lib/aliases*:

  # newaliases

  Some sample entries from */usr/lib/aliases*:

```
#
# Mailing lists for organizations
devel: marshall, swstaff, hwstaff, vlsistaff, service
#
# Mailing lists for special interest groups
pcusers: barryr, cippele, fclark, marshall, locke, rowan
#
# Aliases for people's names
tom:            tjones
#
# Aliases to handle mail to system accounts
root:           tjones
#
# Aliases to handle mail to msgs
msgs: "|/usr/ucb/msgs -s"
#
# Aliases to handle mail to notesfiles
bugs: "|/usr/spool/notes/.utilities/nfmail bugs"
#
# Aliases to handle mail to print queues
printer: "|/usr/ucb/lpr"
#
# Aliases for people at other sites
fsmith:         ihnp4!allegra!convex!convexe!fsmith
```

# USER MAIL (Continued)

- Make sure no "circular" forwarding addresses are in the aliases database on a network
- The *sendmail* utility is used to deliver mail
  - o Mail addresses of all the mail recipients are analyzed
  - o Mail is distributed via UUCP or the Ethernet
- */usr/lib/sendmail.cf* is used to parse these mail addresses
- When *sendmail.cf* is modified:
  - o Create the "frozen" configuration file */usr/lib/sendmail.fc* with the command:

    ```
    # sendmail -bz
    ```
- The *sendmail.cf* distributed by Convex assumes that:
  - o Addresses like `user@sitename` are Internet (Ethernet) addresses and UUCP addresses

# SUMMARY OF FILES USED FOR CREATING USER ACCOUNTS

| FILENAME | DESCRIPTION |
|---|---|
| */etc/passwd* | Contains login information for each user |
| */etc/uidcount* | Determines the user ID number for each added account |
| */etc/group* | Contains additional group names and ID numbers for users |
| */etc/ptmp*<br>*/etc/rtmp*<br>*/etc/vtmp* | Temporary files that are created when *vipw* or *nu* is executed |
| */etc/pwrestrict* | Controls if and when a user changes the password |
| */etc/shells*<br>alle zugelassenen Shells | Controls which shells a user can invoke |
| */usr/skel* | Determines the initial contents of each user's home directory |
| */etc/nurc* | Determine values to be used by the *nu* utility when building new user accounts |
| */usr/lib/aliases* | Contains a set of system-wide aliases used by the *sendmail* utility |
| /tmp/nu.log | log file für nu |

# THE LOGIN PROCESS

- *init* enables each login port in the */etc/ttys* file doing the following:
  - o Forks a new shell
  - o Opens the port for reading and writing
  - o Execs the */etc/getty* program on the port
- */etc/getty* gets information from the */etc/gettytab* file to:
  - o Initialize the port to the correct speed
  - o Display the system banner and login prompt
  - o Read in the login name from the keyboard
- */etc/getty* calls the */bin/login* program with the login name as an argument
- */bin/login* reads in the password and completes the login

# THE LOGIN PROCESS (Continued)

- */bin/login* initializes the environment variables
  - o Obtains information from the */etc/passwd* and */etc/termcap* files
  - o Sets the environmental variables such as PATH, HOME, TERM, SHELL
  - o Displays the*/etc/motd* file

- Shell programs executed by the C-shell at login:

| | |
|---|---|
| *~/.cshrc* | */bin/csh* initialization script |
| */etc/login* | System-wide login sourced by */bin/csh* |
| *~/.login* | User-specific login sourced by */bin/csh* |

- Default *.login* and *.cshrc* files are copied from the */usr/skel* directory

- *.login* and *.cshrc* can be modified to customize the users environment

- To escape *.cshrc* for omitting interactive commands use: `if ($?ENVIRONMENT) exit`

# THE LOGIN PROCESS (Continued)

- */bin/login* initializes the environment variables
  - o Obtains information from the */etc/passwd* and */etc/termcap* files
  - o Sets the environmental variables such as PATH, HOME, TERM, SHELL
  - o Displays the*/etc/motd* file
- Shell programs executed by the C-shell at login:

| | |
|---|---|
| *~/.cshrc* | */bin/csh* initialization script |
| */etc/login* | System-wide login script for */bin/csh* |
| *~/.login* | User-specific login script for */bin/csh* |

- Default *.login* and *.cshrc* files are copied from the */usr/skel* directory
- *.login* and *.cshrc* can be modified to customize the users environment

# CONFIGURING TERMINAL LINES

- Hardware considerations:
  - o Check set-up parameters for the terminal
  - o Connect terminal to an available ACM-001 controller
  - o Create a device file for each terminal line in the */dev* directory
    - * Use */dev/MAKEDEV* script
  - o Add a *tty* entry in the */etc/ttys* file for each added terminal line
  - o See *Filesystem Structure* for more information

# THE */etc/ttys* FILE

- The *init* program uses the */etc/ttys* file to determine which ports should have a *getty* process.
- */etc/ttys* contains the */etc/gettytab* entry name (parameter) for *getty* to use
- *init* passes the entry name as a parameter to *getty*
- Example of */etc/ttys* file:

```
console   "/etc/getty std.9600"     vt100n   on    secure
tty00     "/etc/getty std.9600"     vt100n   on    secure
tty01     "/etc/getty std.9600"     vt100n   on    secure
tty02     "/etc/getty std.9600"     vt100n   on    secure
tty03     "/etc/getty std.9600"     vt100n   on    secure
tty04     "/etc/getty std.9600"     vt100n   on    secure
tty05     "/etc/getty std.9600"     vt100n   on    secure
tty06     "/etc/getty std.9600"     vt100n   on    secure
tty07     "/etc/getty std.9600"     imagen   off   secure
cua0      "/etc/getty D1200"        dumb     off   secure
ttyd1     "/etc/getty D1200"        vt100n   on    secure dial
ttyd2     "/etc/getty D1200"        vt100n   on    secure dial
tty0b     "/etc/getty std.9600"     vt100n   on    secure
tty0e     "/etc/getty std.9600"     vt100n   on    secure
tty0c     "/etc/getty std.9600"     vt100n   on    secure
tty0d     "/etc/getty std.9600"     vt100n   on    secure
tty0f     "/etc/getty std.9600"     vt100n   on    secure
tty10     "/etc/getty std.9600"     vt100n   on    secure
tty11     "/etc/getty std.9600"     vt100n   on    secure
```

# ENABLING AND DISABLING PORTS

- The utilities */usr/convex/on* and */usr/convex/off* enable and disable login ports

- The *on* and *off* command formats are:

  `/usr/convex/on` *options line*
  `/usr/convex/off` *options line*

- These utilities modify the *ttys* file (on/off field) and send a signal to notify *init* of the changes

- The *-n* option modifies *ttys* but does not send a signal to *init*

- To turn off a specific *tty*:

  `# off tty03`

  - o This allows a user to continue working on a line until the current working session terminates
  - o At termination of the session the line is set to *off*
  - o The *off* option *-f* forces immediate logoff

# THE */etc/gettytab* FILE

- *getty* uses */etc/gettytab* to determine:
  - o tty line's characteristics
  - o How to initialize the *tty*
  - o The login message and prompt
- The general format of each entry:
  - o Each entry contains colon-separated fields
  - o All lines (except the first) begin with a colon
  - o All lines end with a colon
  - o All lines (except the last) end with a backslash

*one-character_name*|*symbolic_name*:\
:*attribute*:*attribute*:

- The first field is a one-character entry name
  - o Symbolic names follow separated from each other by a vertical bar
- Other fields describe line attributes
  - o Each attribute has a two-letter name

# */etc/gettytab* ATTRIBUTES

- Some of the common line attributes that are defined in */etc/gettytab*:

| Name | Description |
|---|---|
| `ap` | Any parity |
| `ht` | Hardware tabs |
| `ep` | Even parity |
| `sp#`*speed* | Line speed |
| `im=`*message* | Initial message displayed |
| `lm=`*prompt* | Login prompt |
| `nx=`*next* | Identifies *next* entry to use if */etc/getty* detects a line break |
| `cd#`*msec* | Carriage return delay in milliseconds |
| `fd#`*msec* | Form-feed delay in milliseconds |
| `tc=`*next* | Identifies to continue attribute description at *next* |

- The system's hostname will be substituted for *%h* in the *im* and *lm* fields

```
:im=\r\nThe name of our system is (%h)\n\r:
```

- Default values for some fields are kept in the entry *default*

# */etc/gettytab* EXAMPLE

- */etc/gettytab* **entry examples:**

```
default:\
   :ap:fd#1000:\
   :im=\r\nConvex Systems (%h)\n\r:\
   :sp#1200:
2|std.9600|9600-baud:\
   :sp#9600:
z|std.19200|19200-baud:\
   :sp#19200:
0|d300|Dial-300:\
   :nx=d1200:cd#2:sp#300:
d1200|Dial-1200:\
   :nx=d150:fd#1:sp#1200:
d150|Dial-150:\
   :nx=d110:lm@:tc=150-baud:
d110|Dial-110:\
        :nx=d300:tc=300-baud:
```

# DEFINING TERMINAL TYPES

- The file */etc/ttys* defines the terminal type for each line
- */etc/login* uses */etc/ttys* to set TERM at login time
- Each login line needs an entry in */etc/ttys* with the format:

  *ttyno getty type on-off security*

- Example of */etc/ttys* file:

```
console   "/etc/getty std.9600"     vt100n   on
tty00     "/etc/getty std.9600"     vt100n   on
tty01     "/etc/getty std.9600"     vt100n   on
tty02     "/etc/getty std.9600"     vt100n   on
tty03     "/etc/getty std.9600"     vt100n   on
tty04     "/etc/getty std.9600"     vt100n   on
tty05     "/etc/getty std.9600"     vt100n   on
tty06     "/etc/getty std.9600"     vt100n   on
tty07     "/etc/getty std.9600"     imagen   off
cua0      "/etc/getty D1200"        dumb     off
ttyd1     "/etc/getty D1200"        vt100n   on
ttyd2     "/etc/getty D1200"        vt100n   on
```

# THE */etc/termcap* FILE

- */etc/termcap* defines the specific characteristics and command character sequences for terminal types
- The value of the TERM environment variable should match a *termcap* entry name
- Utilities such as *vi*, *emacs*, and *more* use the value of the environment variable TERM
- *termcap* entries may also be placed in the environment variable TERMCAP
- Sample TERMCAP entry:

```
sx|ansi|ansi terminal:\
        :co#80:li#24:cl=50\E[;H\E[2J:\
        :bs:am:cm=\E[%i%d;%dH:nd=\E[C:up=\E[A:\
        :ce=\E[K:ho=\E[H:pt:
```

- See ConvexOS Programmers Manual System management sec 8, getty(8), gettytab(5), termcap(5)

# SUMMARY OF FILES USED IN THE LOGIN PROCEDURE

| FILENAME | DESCRIPTION |
|---|---|
| */etc/ttys* | Determines which tty lines display a login and type of terminal |
| */etc/gettytab* | Determines user login banner; tty line's characteristics |
| */etc/termcap* | Describes the characteristics of a specific terminal type |
| */etc/login* | System-wide login script |
| *~/.login* | Sets up user's login environment |
| *~/.cshrc* | Sets up C shell environment for individual user |
| */etc/motd* | Displays system-wide message on first login for each user |

# MAGNETIC TAPE SUPPORT

- A Convex machine may connect to a maximum of 32 magnetic tape drives (*unit0* to *unit31*)
- Three tape densities are supported: 800bpi, 1600bpi, and 6250bpi
- When a file open for writing is closed, two end-of-file marks are written
- If the tape is not rewound, it is positioned with the read/write head between the two tapemarks
- A standard tape consists of a series of 1024-byte records terminated by an end-of-file
- A tape dataset is treated like any other file
- When referencing a tape as an ordinary file, the *block* interface is used (*/dev/mt*)
- When dealing with foreign tapes or very long records, the *raw* interface is used (*/dev/rmt*)

# TAPE DEVICES

- Format for tape device names:
  - o Block device: `/dev/mtX`
  - o Raw device: `/dev/rmtX`
  - o *X* is a device number in the range 0-247
  - o For every */dev/mt* device, a corresponding */dev/rmt* device exists
  - o Each physical tape unit supports up to 6 logical tape devices

First Four Tape Units

| Density | Rewind? | unit0 | unit1 | unit2 | unit3 |
|---|---|---|---|---|---|
| 800 bpi | rewind | mt0 | mt1 | mt2 | mt3 |
| 800 bpi | no rewind | mt4 | mt5 | mt6 | mt7 |
| 1600 bpi | rewind | mt8 | mt9 | mt10 | mt11 |
| 1600 bpi | no rewind | mt12 | mt13 | mt14 | mt15 |
| 6250 bpi | rewind | mt16 | mt17 | mt18 | mt19 |
| 6250 bpi | no rewind | mt20 | mt21 | mt22 | mt23 |

Second Four Tape Units

| Density | Rewind? | unit4 | unit5 | unit6 | unit7 |
|---|---|---|---|---|---|
| 800 bpi | rewind | mt32 | mt33 | mt34 | mt35 |
| 800 bpi | no rewind | mt36 | mt37 | mt38 | mt39 |
| 1600 bpi | rewind | mt40 | mt41 | mt42 | mt43 |
| 1600 bpi | no rewind | mt44 | mt45 | mt46 | mt47 |
| 6250 bpi | rewind | mt48 | mt49 | mt50 | mt51 |
| 6250 bpi | no rewind | mt52 | mt53 | mt54 | mt55 |

# TAPE UTILITIES

- Retired tape commands (ConvexOS 7.1)
  - o tpalloc - tape drive allocation
  - o tpc - tape request queue control
  - o tpd - monitoring tape drive
  - o tpdealloc - tape drive deallocation
  - o tpmnt - tape mount request
  - o tpq - print tape request queue and tape unit utilization
  - o tprm - remove jobs from the tape mount request queue
  - o tpumnt - tape unmount request

- New tape commands (ConvexOS 8.X)
  - o tpattr - change attributes used for future labeled tape files
  - o tpconfig - configure the tape system
  - o tplabel - create a new labeled tape
  - o tpmount - mount tape or allocate drive
  - o tpqueue - print tape request queue and tape unit utilization
  - o tpunlabel - remove labels from a labeled tape
  - o tpunmount - unmount tape or deallocate drive
  - o tpwait - wait for a *tpmount* to complete

# LINKING TO TAPE DEVICES

- Establishes a logical link to a device file so that a user can access the tape system
- Allows a user access to a tape device, even if the user does not have permission to access the device directly
- Gives the user exclusive use of a tape unit
- Attempts to find a free tape unit which matches the user's specifications
- Default device characteristics for the mount request are defined via the tape system configuration utility *tpconfig* (ConvexOS 8.X)
- If the *opreq* queueing is enabled, sends the tape mount request to the tape operator
- The ConvexOS 8.X *tpmount* request performs similar functions to the following two ConvexOS 7.1 commands:
  - o *tpalloc* -- allocate a tape drive
  - o *tpmnt* -- request a tape reel to be mounted by an operator

# FEATURES OF ConvexOS 8.X TAPE SYSTEM

- Types of tapes supported
  - o UNIX unlabeled tapes
  - o ANSI standard labeled tapes identified by Volume Serial Numbers (VSNs)
- Tape mount requests
  - o May be handled directly by the tape system
  - o May be passed to an operator if the *opreq* tape queueing is enabled
- Multivolume ANSI labeled tape sets
  - o Are tapes identified as volumes of the same set in the ANSI tape header label
  - o If *opreq* is active, the system automatically requests the operator to switch tapes whenever necessary

# MORE FEATURES OF THE 8.X TAPE SYSTEM

- Optional control of tape drives
  - o Tape system normally controls all drives
  - o A *tpmount* request must precede the use of a tape drive
  - o A tape drive is reserved for the exclusive use of its owner
  - o A tape drive can not be directly accessed unless specific permission has been given in */usr/lib/tape/config.db*
- Automatic tape drive selection
  - o A user does not need to specify a particular drive or device file
  - o The tape system automatically chooses a drive and device which match the characteristics specified by the user
- Automatic Volume Recognition (AVR)
  - o When a labeled tape is placed on a drive, the system reads the VSN
  - o The system matches the tape with the process which issued the *tpmount* command

## *tpdaemon*

- Controls the tape system
- Uses the information defined in the */usr/lib/tape/config.db* file
- Automatically satisfies mount requests for labeled tapes that have been pre-mounted
- Reclaims idle tape drive
- Calls *ansidaemon* to handle ANSI labeled tapes
- Performs AVR (Automatic Volume Recognition)

# TAPE CONFIGURATION DATABASE

- Is contained in the */usr/lib/tape/config.db* file
- Is built by the system administrator using the *tpconfig* utility
- Defines the following conditions for the tape system
  - o Whether a user may access a tape drive without using *tpmount*
  - o Whether label processing may be bypassed
  - o Which tape drives are not controlled by the tape system
  - o Types of label processing recognized by the tape system
  - o Types of tape drives supported
  - o Default drive type, density, speed, and flags for *tpmount*
  - o Enables tape queueing -- all tape mount and drive allocation requests are passed to *opreq* for an operator to handle
  - o Disables tape queueing -- drive allocations are handled by *tpdaemon* and the user hangs his own tapes

# VIEWING CONTENTS OF *config.db*

- The *tpconfig* utility
  - o Functions as a command if arguments are supplied
  - o Acts as a shell and prompts for commands if arguments are not supplied

- Format:

  `tpconfig [`*command*`]`

- Commands which are useful to a user of the tape system:

| | |
|---|---|
| `show all` | Show all the information in */usr/lib/tape/config.db* |
| `show defaults` | Show all the default values for tape drives |
| `show labels` | Show all of the available label types |
| `snapshot [`*file*`]` | Generates an ASCII file which contains the set of *tpconfig* commands which will create a database equivalent to the current *config.db* |

# SAMPLE *config.db*

- **Results of executing *tpconfig show all*:**

```
% tpconfig show all
Database access is read-only.
Drives:
   mt:0      timeout: 60  Controlled
      Alloc access:
         Users:   All
         Groups:  All
      Bypass access:
         Users:   None
         Groups:  None
      Nodes:
      /dev/rmt8     speed:   density: 1600   Rewind      Char
      /dev/rmt12    speed:   density: 1600   No rewind   Char
      /dev/rmt16    speed:   density: 6250   Rewind      Char
      /dev/rmt20    speed:   density: 6250   No rewind   Char
      /dev/mt8      speed:   density: 1600   Rewind      Block
      /dev/mt12     speed:   density: 1600   No rewind   Block
      /dev/mt16     speed:   density: 6250   Rewind      Block
      /dev/mt20     speed:   density: 6250   No rewind   Block

Labels:
   ansi daemon: /usr/lib/tape/ansidaemon

Queueing is: Disabled

Defaults:
   Density: 6250
   Speed: No default
   Drive type: mt
   Mount flags: Character special, No-rewind
```

# *tpmount* COMMAND

- If operator queueing is disabled, *tpmount* is used to request that *tpdaemon* allocate a tape drive

- If operator queueing is enabled, *tpmount* is used to pass all requests to *opreq* so that the operator may allocate the tape drive and mount the tape

- *tpmount* will create a symbolic link in the current directory
  - o The *-s name* option is used to supply a name for the symbolic link
  - o If the *-s* option is omitted, the symbolic link is named *TAPE*

- *tpmount* format:

  % tpmount *options*

- Format of *tpmount* options (*b*, *q*, *r*, *R*):

| | |
|---|---|
| *-option* | Turn on desired option |
| *+option* | Turn off desired option; deselect option if it is currently selected |

# OPTIONS FOR UNLABELED TAPES

- Options used with unlabeled tapes:

| | |
|---|---|
| `-a` *dev* | Requests unit of device name *dev* |
| `-d` *bpi* | Requests unit with density *bpi* |
| `-i` *ips* | Requests unit with speed *ips* |
| `-m` *mode* | Device mode (*block*, *char*, or *label*) If queueing is disabled and *-m label* is used, the tape must be placed on the drive before executing *tpmount* |
| `-q` | Causes a drive allocation to be queued; use when queueing is disabled |
| `+q` | Deselects the *-q* option |
| `-r` | Requests unit with rewind on close (only unlabeled tapes) |
| `+r` | Deselects the *-r* option |
| `-R` | Tape should be read-only (mount without a write ring) |
| `+R` | Deselects the *-R* option |
| `-t` *typ* | Requests unit of drive type *typ* (defined in *tpconfig.db*) |
| `-s` *name* | Requests unit with a symbolic link to *name*; must be the last option |

# OPTIONS FOR LABELED TAPES

- Options used with labeled tapes:

| | |
|---|---|
| `-b` | Bypass label processing so that a labeled tape may be used in the block or character device modes |
| `+b` | Deselects the *-b* option |
| `-c` *comment* | Comment to be sent to the operator; enclose phrase in quotes |
| `-f` *N* | Number of files to skip after mounting the tape device (only labeled tapes) |
| `-l` *label_typ* | Specified label type; only valid type is *ansi* |
| `-m` *mode* | Device mode (*block*, *char*, or *label*) |
| `-s` *name* \ `[vsn1,...]` | Requests a unit with a symbolic link to *name* The *name* is followed by a list of VSN's which compose one ANSI fileset If VSN's are listed on an unlabeled file, they are treated as comments |

# *tpwait* COMMAND

- Waits for a *tpmount* request to complete
- Is used with the *tpmount -B* command which places the *tpmount* request in the background
- Format of *tpwait*:

  `tpwait [-s` *link_name*`]`

- The *-s* option is required only if more than one outstanding tape mount request exists

# *tpunmount* COMMAND

- Used to unmount a tape or deallocate a tape drive
- The *tpunmount* format:

```
% tpunmount [-k] [-s name]
```

- The *-k* option:
  - o Keeps the tape system from taking the tape offline
  - o Do not use when *opreq* is enabled
  - o The *-k* option is included only for backward compatibility
- The *-s* option:
  - o Defines the tape to be unmounted or the drive to be deallocated
  - o The *-s* option is not required if only one tape is mounted or only one drive is allocated

```
% tpmount -s MYTAPE
% tpunmount
```

  - o The *name* supplied may also be a true device name such as */dev/rmt20* even if the link file (e.g. *TAPE*) exists

# *tpqueue* COMMAND

- Prints tape request queue on standard output
- Format:

```
tpqueue [-l]
```

- The *-l* option generated a detailed listing
- Using *tpqueue* to monitor the tape system:

```
% pwd
/tmp
% ls
% tpmount
Tape device /dev/rmt20 allocated.
% ls
TAPE
% tpqueue -l
Drive  Tape Device  User Device  User   VSN   Symbolic Link
mt:0   /dev/rmt20   /dev/rmt20  user   None  /tmp/TAPE
% tpmount -q -B -s TAPE2
% tpqueue -l
Drive  Tape Device  User Device  User   VSN   Symbolic Link
mt:0   /dev/rmt20   /dev/rmt20  user   None  /tmp/TAPE
                                user   None  /tmp/TAPE2
% tpunmount -s TAPE2
% tpqueue -l
Drive  Tape Device  User Device  User   VSN   Symbolic Link
mt:0   /dev/rmt20   /dev/rmt20  user   None  /tmp/TAPE
% tpunmount
% tpqueue -l
Drive  Tape Device  User Device  User   VSN   Symbolic Link
%
```

# TAPE SYSTEM CAVEATS

- A user *must* use what he allocates, and *must* allocate a drive before using it
- Timeouts are measured with respect to the allocated drive
- Tape system is not integrated with CXbatch
- Is difficult to determine the VSN of a tape without a paper label
- Amount of data which can be stored on a tape varies due to the following:
  - o Tape density
  - o Blocksize
  - o Tape length
- Tape librarian is not available
- Asynch I/O can not be performed on labeled tapes
- The *ansitar* utility will not work with tapes which were mounted as labeled

# MAGNETIC TAPE MANIPULATION

- Used to give commands to a magnetic tape drive
- Must be used on a *raw* device
- Format of *mt*:

  `mt [-f` *tapename*`]` *command* `[`*count*`]`

- The *tapename* is the link to the actual device
  - o Is the symbolic name used in the *tpmount* command
  - o If absent, is assumed to be *TAPE*
  - o If *TAPE* does not exist, then the device used is */dev/rmt12*
- *mt* options:

| | |
|---|---|
| `bsf` | Back space *count* files<br>`% mt bsf 2` |
| `bsr` | Back space *count* records<br>`% mt -f TAPE bsr 5` |
| `eof, weof` | Write *count* end-of-file marks at the current position on the tape<br>`% mt -f /dev/rmt12 eof` |

# THE *mt* OPTIONS

- *mt* options (cont.):

| | |
|---|---|
| `fsf` | Forward space *count* files<br>`% mt -f /dev/rmt8 fsf 3` |
| `fsr` | Forward space *count* records<br>`% mt -f TAPE fsr 7` |
| `gap` | Write *count* extended inter-record gaps at the current position on the tape<br>`% mt -f TAPE gap 2` |
| `offline` | Rewind the tape and place the tape unit offline<br>(*count* is ignored)<br>`% mt -f TAPE offline` |
| `rewoffl` | Rewind the tape and place the tape unit offline<br>(*count* is ignored)<br>`% mt -f TAPE rewoffl` |
| `status` | Print status information about the tape unit<br>0: operation successful<br>1: unrecognizable command<br>2: operation failed<br>`% mt -f MYTAPE status` |

# EFFECTS OF *mt* COMMANDS

| The *mt* Command | Unlabeled Mode (Character) | Labeled Mode |
|---|---|---|
| `eof` *n*, `weof` *n* | Write *n* EOF marks | Write *n* empty ANSI files |
| `fsf` *n* | Forward space *n* files | Forward space *n* logical files |
| `fsr` *n* | Forward space *n* records | Forward space *n* logical records |
| `bsf` *n* | Back space *n* files | Back space *n* logical files |
| `bsr` *n* | Back space *n* records | Not allowed |
| `rewind` | Rewind tape | Rewind tapeset; stop at first ANSI file |
| `offline`, `rewoffl` | Take tape offline | Not allowed; use *tpunmount* |
| `gap` *n* | Write *n* inter-record gaps | Command ignored |
| `status` | Return tape status | Return physical tape status |

# THE */etc/dump* UTILITY

- The */etc/dump* utility is used to backup file systems
  - o Full dumps are entire file systems
  - o Incremental dumps are partial file systems
- Dumps should only be done on inactive file systems
- Incremental dumps are done by levels
  - o The *u* option is specified
  - o Files not modified since the previous lower-level dump will not be dumped
- The *dump* command format:

  `/etc/dump` *level file_system*
  `/etc/xdump` *level file_system*

- *level* can be 0 through 9
- To record the dumpdate include the *u* option

```
# /etc/dump 0u /dev/rda0a
```

# INCREMENTAL DUMPS

- The *u* option is specified to record the date and dump level in the file */etc/dumpdates*
- *dump* looks at *dumpdates* and only dumps files modified after the last date of a lower level dump
- The format of a *dumpdates* entry:

  *file_system level date*

- Example *dumpdates* file:

```
/dev/rst3    0 Fri Jul 20 12:40:11 1990
/dev/rst3    1 Sat Jul 21 12:46:52 1990
/dev/rst3    2 Mon Jul 23 08:40:10 1990
/dev/rst3    3 Wed Jul 25 10:45:57 1990
/dev/rst3    4 Thu Jul 26 12:18:02 1990
/dev/rst3    0 Fri Jul 27 09:59:41 1990
/dev/rst3    1 Sat Jul 28 10:00:58 1990
```

# DEFAULT DUMP DEVICE

- Options can be specified with *level* to:
  - o Modify the dump device name
  - o Inform *dump* of any changes to the default

- The defaults are:
  - o Dump device */dev/rmt8* at 1600 bpi
  - o This can be changed with the *G* option to */dev/rmt16* and 6250 bpi

    Example:

```
# /etc/dump 0Gu /dev/rda1h
```

- The default tape size is 2300 feet This can be changed with the *s* option
  - o Example:

```
# /etc/dump u0s 1200 /dev/rda0h
```

# VERIFYING DUMP TAPES

- Dump tapes can be verified for tape I/O errors
- Verify does not guarantee the correctness of the directories and files contained in the dump file
  - o /etc/restore -V vergleicht nicht auf Korrektheit sondern nur auf Lesbarkeit
- The sure way to prove the files is to read them back onto the disk
  - o /etc/restore -x
- To display the file names on the dump tape
  - o /etc/restore -t

# RESTORING FROM DUMP TAPES

- The */etc/restore* utility restores files from dump tapes
- Either the entire dump or portions of the tape can be extracted
- Files will be restored to the current directory
- File ownership, permissions, and modification times are retained
- Use of restore is determined by the following options:

  `x` Extract files; If no files are specified, all files are extracted
  `i` Extract interactively
  `t` Display the filenames on the dump tape

  Example:

```
# /etc/restore x /smith
```

# INTERACTIVE RESTORE

- Files can be restored interactively if the *i* option is given. The prompt is:

  restore >

- The directories on the tape can be searched
- Files or directories can be marked for extraction
- Marked files are pulled off the tape all at one time
- The commands available for interactive mode are:

| | |
|---|---|
| `add` | Add a file to the extraction list |
| `delete` | Remove a file from the extraction list |
| `ls` | List a directory's contents |
| `pwd` | Display the current directory name |
| `cd` | Change the current directory |
| `extract` | Extract the marked files |
| `quit` | Exit *restore* |
| verbose | use verbose mode |

# SAMPLE INTERACTIVE RESTORE

- **Sample interactive extraction script:**

```
# /etc/restore -iG -> auj 6250 BPI-Band
                      un rewind
restore > ls
 coopride/    darnell/     lost+found/
restore > cd coopride
restore > ls
 .cshrc    .login    .msgsrc  foo      foo.f
 .exrc     .logout   a.out    foo.
restore > add foo foo.f
restore > extract
You have not read any tapes yet.
Unless you know which volume to use, you should start
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
restore > quit
#
```

# TAPE ARCHIVER

*tar*(1)

- Saves and restores multiple files on a single file; can not span tape reels
- Format of *tar*:

  **tar** *option_string* [*name* ...]

- *tar* options:

  c Create a new tape; the files are added at the beginning of the tape
  % `tar c file1 file2`

  r The named files are written on the end of the tape
  % `tar r file1 file2 file3`

  t The specified files are listed each time they occur on the tape. If no file argument is given, all of the files on the tape are listed.
  % `tar t`

  v The verbose option makes *tar* list each file it processes preceded by its function letter. When used with the *t* option, *tar* provides even more information.
  % `tar tv`

# THE *tar* OPTIONS

- *tar* options (cont.):

  x Extract the named files from the tape. If no file argument is designated, the entire contents of the tape is extracted. If the file argument is a directory found on the tape, the entire directory is extracted. The owner, modification time, and mode are restored whenever possible. If multiple copies of the same file are found on the tape, the last copy overwrites all earlier versions extracted from the tape.
  % `tar x source.dir`

  u The named files are added to the tape if (1) they are not present on the tape, or (2) they have been modified since last written on the tape.
  % `tar u file1 file2`

  o Suppress the owner and modes of directories placed in the archive in order to avoid the *"<name>/: cannot create"* error
  % `tar vro file1 file2`

# MORE *tar* OPTIONS

- *tar* options (cont.):

  p Restore files to their original modes, ignoring the present *umask*
  ```
  % tar xvp source.dir
  ```

  *n* Modifier from *0* to *9* which selects an alternate drive on which to mount the tape; the default is drive zero at 1600bpi, which is normally */dev/rmt8*
  ```
  % tar vr1 file1 file2
  ```

  f *Tar* uses the next argument as the name of the archive instead of */dev/rmt?*. If the file is named -, tar writes to standard output or reads from standard input. Thus *tar* can be used in a filter chain.
  ```
  % tar cvf /dev/rmt16 newdir1
  % cd olddir; tar cvf - . | \
  (cd newdir; tar xvf -)
  ```

  w *Tar* prints the action to be performed and then waits for user conformation; if a word beginning with a *y* is entered, the action is executed
  ```
  % tar cvw file1 file2
  ```

# THE *tar* BLOCKING OPTIONS

- *tar* options (cont.):

  b — *Tar* uses the next argument as the blocking factor for tape records; the default is *20*. This option should only be used with *raw* magnetic tapes. The block size is determined automatically when reading tapes.

  ```
  % tar cvb 10 file1
  ```

  B — Forces both input and output blocking to *20* records per block

  ```
  % tar cvB file1 file2
  ```

  l — Requests that *tar* complain if it cannot resolve all the links to the files being dumped; the default is not to print any error messages

  ```
  % tar cvl dir1 dir2 dir3
  ```

  m — Requests that the modification times not be restored; the modification time will be the time of extraction

  ```
  % tar xvm file1 file2
  ```

# THE *tar* DIRECTORY OPTIONS

- *tar* options (cont.):

  h Forces *tar* to follow symbolic links as if they were normal files or directories; normally, *tar* does not follow symbolic links

  ```
  % tar cvh file1 file2
  ```

  -C If a file name is preceded by *-C*, *tar* will change to that directory. This allows multiple directories to be archived using short relative path names.

  ```
  % tar cv -C /usr include -C /etc
  ```

# THE *ansitar* COMMAND

*ansitar*(1)

- Read and writes ANSI multifile labeled tapes
- Will *not* work with tapes which were mounted as ANSI labeled
- Useful for exhange of ASCII character files with a non-Unix system
- Will read VMS *copy* tapes which have been initialized and mounted as VMS volumes with labels
- Format of *ansitar*:

  ansitar [*subcommand*] [*option_values*] [*name* ...]

- A *subcommand* is a concatenation of characters from the following groups -- only one from each group

  [crtx], [bBDF], l, U, v, V, [f]

- Arguments are evaluated in the order in which the associated *subcommand* characters appear

# THE *ansitar* BASIC OPTIONS

- First subcommand group:

  c Create a new tape; the files are added at the beginning of the tape
  ```
  % ansitar c file1 file2
  ```

  r The named files are written on the end of the tape
  ```
  % ansitar r file1 file2 file3
  ```

  t The specified files are listed each time they occur on the tape. If no file argument is given, all of the files on the tape are listed. If both the *t* and *v* options are given, *ansitar* provides even more detailed information about the files.
  ```
  % ansitar t
  ```

  x Extract files from the tape. The * wild card character may be used to get a number of related files from the tape, but it must be quoted to prevent the shell from attempting to expand the wild card.
  ```
  % ansitar x 'file*'
  ```

# THE *ansitar* HEADER OPTIONS

- Second subcommand group:

  b Use next argument as a tape block size
  % `ansitar b 1000 file1`

  B Use next argument as a tape block size and the following argument as the line size (for blocking/unblocking of fixed-length records)
  % `ansitar B 1000 50 myfile`

  D Reading: not required for system level 3 tape with valid HDR2 label; for system level 1 and 2, use next argument as tape block size
  Writing: write HDR2/EOF2 labels and variable-length records (D format, system level 3) with block size of 2048 bytes and maximum line size of 255
  Must be used if writing a tape to be read on the VAX
  % `ansitar xD 2048 file1`

  F Reading: not required if tape have valid HDR2 label; use *B* option for ANSI system level 1 tapes
  Writing: same as option *B*, but write HDR2/EOF2 label for automatic unblocking (F format, system level 3)
  % `ansitar cF 1000 'file*'`

# THE *ansitar* VOLUME OPTIONS

- Other options:

  l Use next argument as a nonstandard label size
  % ansitar cl 133 file1

  U Perform case conversion on alphabetic characters in file names: uppercase to lowercase from tape to Unix, lowercase to uppercase from Unix to tape
  % ansitar cU file1 file2

  v Use the *verbose* mode; generates detailed information concerning labels
  % ansitar cv file1 file2

  V Use the next argument (up to 6 alphanumeric characters) as the tape *vsn*; it is preferable to use only uppercase alphabetics in the argument
  % ansitar cV TAPE00 file1

  f Use the next argument as the tape unit to use; the default is */dev/rmt8*
  % ansitar cf /dev/rmt16 file1

# FURTHER STUDY

- ConvexOS Tape System Guide (710-003130-000)

- Man pages for basic tape operations
  - o tpmount, tpunmount - mount and unmount device or reel
  - o tplabel, tpunlabel - handle tape labels
  - o tpconfig - configure tape system
  - o tpattr - change tape attributes
  - o tpqueue - print tape request queue
  - o tpwait - wait for *tpmount* to complete

- Man pages for control of tape reel
  - o mt - manipulation of tape reel
  - o mtio - magnetic tape interface

- Man pages for copy utilities
  - o cp - copy command
  - o dd - copy and convert data
  - o tar - tape archive command
  - o tar(5) - tar format
  - o ansitar - ANSI tape archive

# SPECIAL FILES

- Special files allow devices to be accessed as files
- The inode for special files contain:
  - o Type - block or character device
    - * Character special files support a *'raw'* unbuffered interface
    - * *Block* special files support a buffered interface
  - o The major device number
  - o The minor device number
- Internally, each device on a UNIX system is identified by a major device number, a minor device number and a device type
- Some devices (disks, tapes) use several minor device numbers for each physical unit
- The list of the devices is contained in /dev
- To obtain a list of the devices and their permissions try the command: *ll -a /dev*

# SPECIAL FILE CREATION

- The *MAKEDEV* shell script (*/dev* directory) creates the *SPECIAL FILES* that allow ConvexOS to communicate with the I/O world

- The *MAKEDEV* command creates all the files required to work with each device type

- The format of the command:

```
cd /dev
MAKEDEV device_IDdevice#
```

- Example:
  - To make device files for the second disk on a Multibus:

```
cd /dev
MAKEDEV da1
```

- The driver identifier type can be found at the beginning of the *MAKEDEV* shell script

# CONVEX UNIX DEVICE TYPES

## The 'real' devices supported in CONVEX UNIX are:

| Controller Name | Driver Name | Device Description | Sample Unit Name |
|---|---|---|---|
| ACM-001 | ca | 16-line async I/O controller | /dev/tty12 |
| -- | cons | System console | /dev/console |
| -- | ct | SPU Cartridge Tape Drive | -- |
| DKC-001 | da | Multibus disk controller | /dev/da0[a-h] |
| DKC-203 | dd | VMEbus disk controller | /dev/dd0[a-h] |
| GPI-001 | dm | 16-bit parallel interface | /dev/dm1 |
| LAN-001 | ex | Ethernet interface | -- |
| PRC-001 | pa | Lineprinter interface | /dev/lp1 |
| VER-001 | pb | Versatec plotter interface | /dev/pb1 |
| MTC-001 | ta | Magtape controller | /dev/mt1 |

## The Pseudo-devices supported in CONVEX UNIX are:

| | | | |
|---|---|---|---|
| -- | pty | Pseudo-teletype | /dev/ptyp1<br>/dev/ttyp1 |
| -- | tt | User's terminal | /dev/tty |
| --- | st | Striped file system | /dev/st1 |
| -- | -- | Swapping space | /dev/drum |
| -- | -- | Kernel virtual memory | /dev/kmem |
| -- | -- | Kernel physical memory | /dev/mem |
| -- | -- | IOP memory | /dev/ccu[3-7] |
| -- | -- | Data sink | /dev/null |

# DEVICE FILES

- Device File Entries In */dev*:

| | |
|---|---|
| *console* | Interface to the system console |
| *cua** | Interfaces to auto-dialing modems |
| *da** | Block-mode interfaces to disk partitions |
| *ccu** | Interfaces to IOP memory |
| *kmem* | Interfaces to kernel virtual memory |
| *lp** | Interfaces to line printers |
| *mem* | Interface to kernel physical memory |
| *mt** | Block-mode interfaces to mag tape drives |
| *null* | The 'bit bucket' |

# DEVICE FILES (page 2)

- Device File Entries In */dev*:

| | |
|---|---|
| *ptyp** | Interfaces to virtual terminals (master) |
| *rda** | Raw-mode interfaces to disk partitions |
| *rmt** | Raw-mode interfaces to mag tape drives |
| *rst** | Raw-mode interfaces to striped file systems |
| *st** | Block-mode interfaces to striped file systems |
| *tty* | Interface to the generic terminal device |
| *tty** | Interfaces to hard-wired terminal lines |
| *ttyd** | Interfaces to dial-up terminal lines |
| *ttyp** | Interfaces to virtual terminals (slave) |

- Use the Shell script */dev/MAKEDEV* for making the device files in */dev*

# ADDING TERMINAL DEVICES

- Format:

  /dev/tty*XY*

  - o *X* is the controller ordinal [0-7]
    - * If *X* is a 'd', then the line is a dialup
  - o *Y* is the line ordinal [0-f]

- To add a new terminal device:

```
# cd /dev
# MAKEDEV ca#ID#
```

- The appropriate 16 *tty* devices are created:

```
crwx-w--w-  1 root  3,  0 Nov 21 01:45 /dev/tty00
crwx-w--w-  1 root  3, 15 Nov 21 01:45 /dev/tty0f
```

- Add an entry for each tty created to the */etc/ttys* file

# CREATING PSEUDO-TERMINAL DEVICES

- Format:

  /dev/pty*XY* (master)

  /dev/tty*XY* (slave)

- *X* is a 1-character group ordinal [e-t]

- *Y* is a 1-digit hex number [0-f]

- Pseudo-teletypes support remote login and the *script* command

- Each pseudo-teletype supports a master side and a slave side

- To create a new pseudo device:

  # cd /dev

  # MAKEDEV pty*ID_#*

- The pseudo devices are created:

```
crw-rw-rw-  1 root     10,   0 Nov 21 02:15 /dev/ptyp0
crwx-w--w-  1 fuka      9,   0 Nov 21 02:15 /dev/ttyp0
```

- Add an entry for each tty created to the */etc/ttys* file

# DISK UNIT NAMES

- Format:
  - /dev/da*XY* (block)
  - /dev/dd*XY* (block)
  - /dev/rda*XY* (raw)
  - /dev/rdd*XY* (raw)
  - o *X* is a decimal unit number [0-9,10-99]
  - o *Y* is a 1-letter partition ordinal [a-h]
  - o Each disk unit is divided into the logical partitions

- To add disks, use:

```
# cd /dev
# MAKEDEV daID# or MAKEDEV ddID#
```

- Eight block and raw devices are created:

```
brw------- 1 root    5,  7 Jan 10 1985 /dev/da0h
crw------- 1 root    5,  7 Apr  4 1985 /dev/rda0h
brw------- 1 root   28,  7 Jan 10 1985 /dev/dd0h
crw------- 1 root   28,  7 Apr  4 1985 /dev/rdd0h
```

- Special disk files are required for striped file systems. A block and raw device file is created for each stripe:

```
# cd /dev
# MAKEDEV stID#
brw------- 1 root    7,  7 Jan 10 1985 /dev/st0
crw------- 1 root   12,  7 Apr  4 1985 /dev/rst0
```

# DISK PARTITIONS

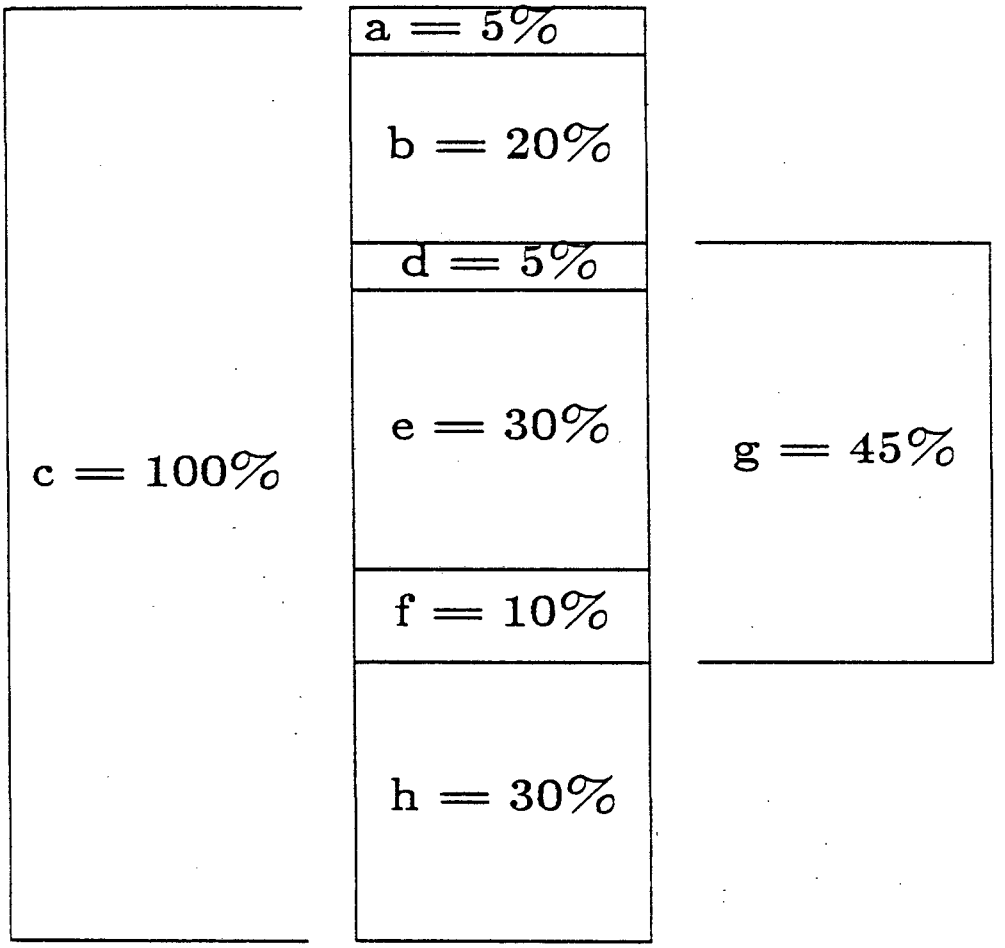
- Physical disks are accessed through disk partitions
- Each partition is a different area of the disk
- Disk partitions are accessed through the disk special files and are named '*a*' - '*h*'
- Disk partitions are used for file systems and swap space

Partition Structure

| Name | Percentage | Description |
|---|---|---|
| a | 5 | Used for / on da0 or dd0 |
| b | 20 | Default on da0 for swap |
| c | 100 | Uses entire disk |
| d | 5 | |
| e | 30 | |
| f | 10 | |
| g | 45 | Combines partitions d + e + f; d,e,f partitions are not available when g is used |
| h | 30 | |

- The max number of partitions per disk is six

# DISK PARTITIONS (page 2)



# THE */etc/disktab* FILE

- */etc/disktab* describes the disk partitions
- *disktab* Entries are in *termcap* format
- *disktab* entries should not be modified
- Some of the field types are:

| Name | Type | Description |
|---|---|---|
| ns | num | Sectors per track |
| nt | num | Tracks per cylinder |
| nc | num | Total cylinder count o |
| b[a-h] | num | Partition block size |
| f[a-h] | num | Partition fragment siz |
| p[a-h] | num | Size of partition |
| se | num | Sector size in bytes |
| ty | str | Type of disk (e.g. rem winchester) |

# */etc/disktab* page 2

- Sample *disktab* entries:

```
dkd-001|dkd001|DKD-001|DKD001|eagle|Eagle:\
     :ty=winchester:ns#45:nt#20:nc#842:rm#3900\
     :pa#37800:ba#8192:fa#1024:\
     :pb#151200:pc#756000:\
     :pd#37800:bd#8192:fd#1024:\
     :pe#227700:be#4096:fe#512:\
     :pf#75600:bf#4096:ff#1024:\
     :pg#341100:bg#4096:fg#512:\
     :ph#225900:bh#4096:fh#1024:
dkd-005|DKD-005|dkd-105|DKD-105|nec:\
     :ty=nec:ns#60:nt#19:nc#760:rm#3070\
     :pa#42496:ba#8192:fa#1024:\
     :pb#170368:pc#849664:\
     :pd#42496:bd#8192:fd#1024:\
     :pe#255488:be#4096:fe#512:\
     :pf#85120:bf#4096:ff#1024:\
     :pg#383360:bg#4096:fg#512:\
     :ph#253312:bh#4096:fh#1024:
```

# SETTING UP THE DISK SYSTEM

- Issues:
  - o Physical resources
  - o Speed and efficiency
  - o Tunable parameters

# BUFFER CACHE

- The larger the buffer cache, the better your disk performance

- Memory used for the buffer cache comes from the general pool of memory to:
  - o Dynamically balance the amount of memory used as the file system cache versus the amount of memory used for programs
  - o Provide cache consistency between mapped file objects and the file system
  - o Allow for Read Ahead / Write Behind capability to speed I/O

- The maximum amount of memory allocated for the buffer cache is the smaller of:
  - o 90% of the amount of memory available at boot time
  - o 8192 times the size of a file system buffer, with a maximum limit of 512 megabytes

# LOAD BALANCING

- Use appropriate fragment and block size for file systems
- Stripe disks to distribute file systems across multiple disks
- Best configuration of resources:
  - o Every physical disk on a separate Multibus/VMEbus
  - o Stripe across discs on separate controllers
  - o Bottlenecking occurs when striping across disks on the same controller
  - o Balance activity on each disk
  - o Avoid multiple active file systems on one disk

# DETERMINING SWAP SPACE

- Swap space is utilized when data is moved from memory to disk
- By default, *da0b* is used for swap space
  - o Swapping is allowed on additional *b* partitions (up to 16 disks)
- Determining swap space requirements:
  - o For configuration purposes, 75% of available memory plus swap space, must be greater than the virtual size of the sum of all processes
  - o For reasonable throughput most processes should fit in real memory
- Consider the size of the largest set of processes that will be run
- Consider the sum of available real memory plus swap space to determine the size of the largest set of processes that can be run
- System memory requires about 16 Mbytes of physical memory
- File systems cannot share a swap partition

# BLOCK AND FRAGMENT SIZES

- Fragments and blocks are contiguous
- Fragment size specifies the minimum disk space used by a file
- Large block size implies fewer disk operations - faster transfer
- Small fragment size implies less wasted space for very small files
- Adjust block and fragment size depending on size of files and activity
  - o Allowable block sizes are: 4k, 8k, 16k, 32k, 64k
  - o Usually fragment size is set of block size/8
  - o Allowable fragment sizes are: block size, 1/8, 1/4, or 1/2 block size
- See: A Fast file System for Unix page 4 (Tutorial papers)

# CREATING FILE SYSTEMS OVERVIEW

- Determine which partitions are available
  - o Use the *df* command
  - o Use the */etc/getst* command
  - o View the */etc/fstab* file
- Shutdown to single-user

```
# /etc/shutdown now
```

- Determine if a backup is necessary
  - o Do a full dump of the appropriate file system(s)

```
# /etc/dump 0uO /file_system
```

- Unmount file system(s) that are to be modified or changed

```
# /etc/umount /file_system
```

- Modify the */etc/fstab* file
  - o Supply the device name, mount point, type of file system, file system attributes
- Determine block/fragment size for each file system

# CREATING FILE SYSTEMS OVERVIEW (cont.)

- Create the file system with:

  # /etc/newfs *raw_device disktype*

- Make directory for file system (mount point)

  # mkdir */file_system_name*

- Mount the file system

  # /etc/mount *blk_device /file_system_name*

- Change access permission if necessary

- Run a file system check (*/etc/fsck*) on unmounted file system (raw device)

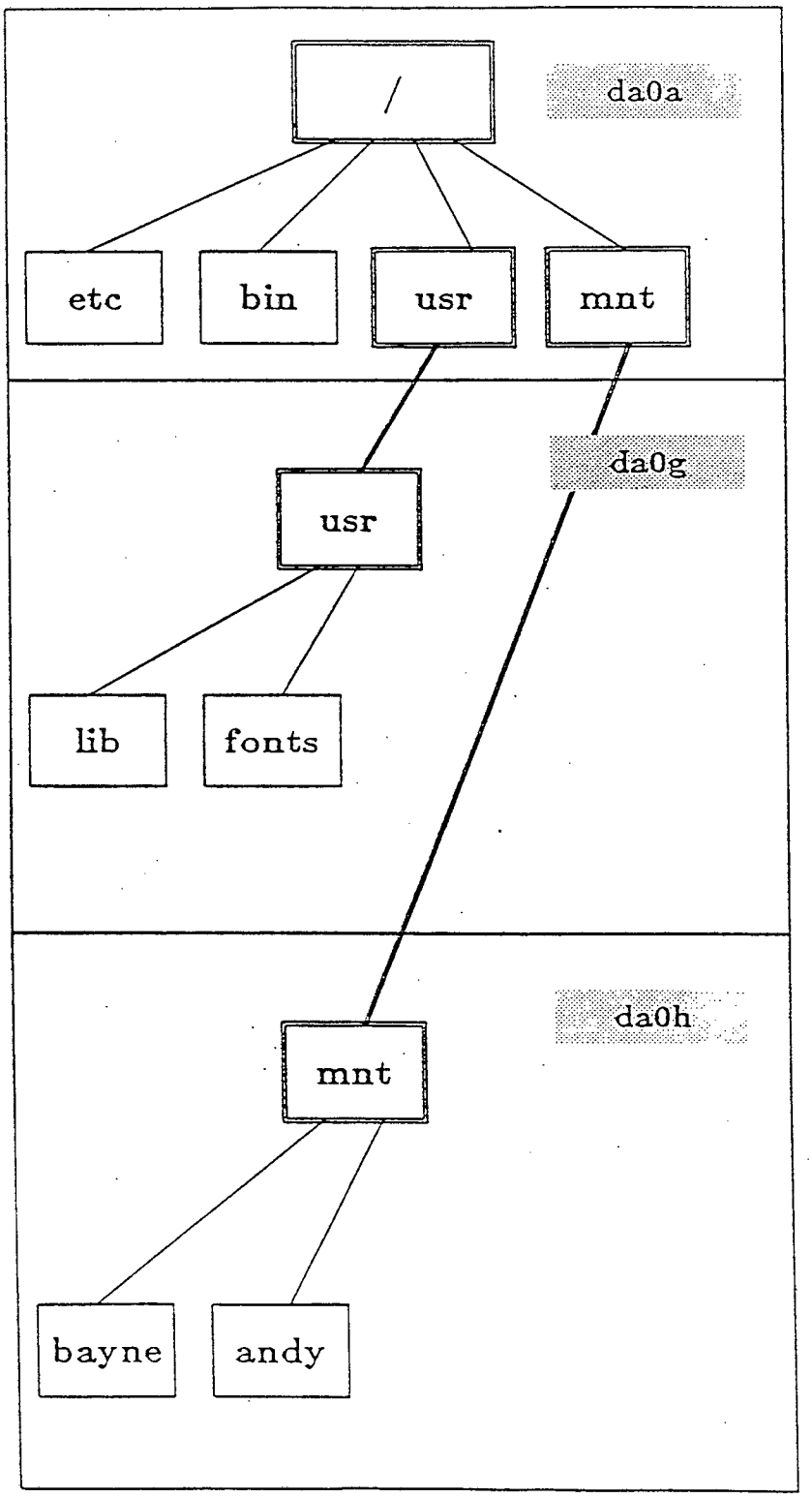
  # /etc/umount */file_system_name*
  # /etc/fsck /dev/*raw_device_name*

- Return to multi-user mode (^D)

# MOUNTING FILE SYSTEMS

- File systems are combined to make the directory structure the user sees
  - o This is done through mounting
- The root file system is mounted during the boot process
  - o The root directory is known as the / directory
- File systems are mounted on directories called mount points which point to disk partitions
- References to a directory which is mounted refer to the disk partition mount point such as *rda0f*

# MOUNTING OVERVIEW



# THE */etc/umount* COMMAND

- The */etc/umount* command unmounts file systems
- When creating a file system, make sure that no file system is mounted on the partition to be used
- The *umount* command format is:

  `/etc/umount` *options /file_system*

- If *umount* is invoked with the *-a* option, all eligible file systems (those specified in */etc/fstab*) will be unmounted
- Example to unmount one file system:

```
# /etc/umount /mnt
```

- Example to unmount all file systems listed in the *fstab* file:

```
# /etc/umount -a
```

- File systems containing open files cannot be unmounted

# THE */etc/fstab* FILE

- */etc/fstab* describes the file system and swap partitions
- *fstab* entries are used by utilities that need file system information, such as *mount*, *umount* and *fsck*
- *fstab* entries are created with the format:

  *device_name directory type options frequency passnum*
- Valid *types* are:

| | |
|---|---|
| `4.2` | Berkeley 4.2 UNIX file system. |
| `nfs` | NFS file system. |
| `swap` | Swap partition. |
| `ignore` | Ignore this record. |

# */etc/fstab* page 2

- *options* contain comma-separated option words
- The following *options* are good for all file system types:

| | |
|---|---|
| `rw` | Read/write. |
| `ro` | Read-only. |
| `suid` | Set-uid execution allowed. |
| `nosuid` | Set-uid execution not allowed. |
| `quota` | Usage limits enforced. |
| `noquota` | Usage limits not enforced. |
| `hide` | Ignore with the *mount -a* command. |

- *frequency* indicates how often file systems should be dumped to tape
- *passnumber* indicates ordering for *fsck*
- Sample *fstab* entries:

```
/dev/da0a / 4.2 rw 1 1
/dev/da0b (swap_space_1_of_3) ignore xx 0 0
/dev/da0c (whole_disk) ignore xx 0 0
/dev/da0d /mnt1 4.2 rw 1 1
/dev/da0e /usr 4.2 rw 1 1
/dev/da0f /rev0/usr 4.2 rw 1 1
/dev/da0g (partitions_0d/0e/0f) ignore xx 0 0
/dev/da0h (empty) ignore xx 0 0
/dev/st8 /scratch 4.2 rw 1 1
argos:/att /argos/att nfs soft,bg 0 0
```

# USING *newfs*

- File systems are created on disk partitions with the */etc/newfs* command
- *newfs* command format:

  /etc/newfs *options raw_device disktype*

- Some of the common *options* are:

| | |
|---|---|
| -b | *block_size* |
| -f | *fragment_size* |
| -m | *free_space_%* |
| -i | *bytes/inode* |

- The file system is created on partition *raw_device*
  - o *disktype* names an entry in */etc/disktab*
  - o The *disktype* can be viewed in the *spu* file */ioconfig* disk
- Examples:

```
# /etc/newfs /dev/rda0d dkd-005
# /etc/newfs -b 64k -f 8k -m 5 -i 8192 /dev/rda0d dkd-005
# /etc/newfs -b 16k -f 2k /dev/rda0d dkd-005
```

# THE */etc/mount* COMMAND

- The */etc/mount* command mounts file systems
- *mount* command format:

  `mount` *options blk_device /mount_point*

- *blk_device* is the partition on which the file system resides
- *mount point* is the directory on which to mount the file system
- Example:

```
# /etc/mount /dev/da1h /mnt
```

- If *mount* is invoked with the *-a* option, all eligible file systems (from */etc/fstab*) will be mounted

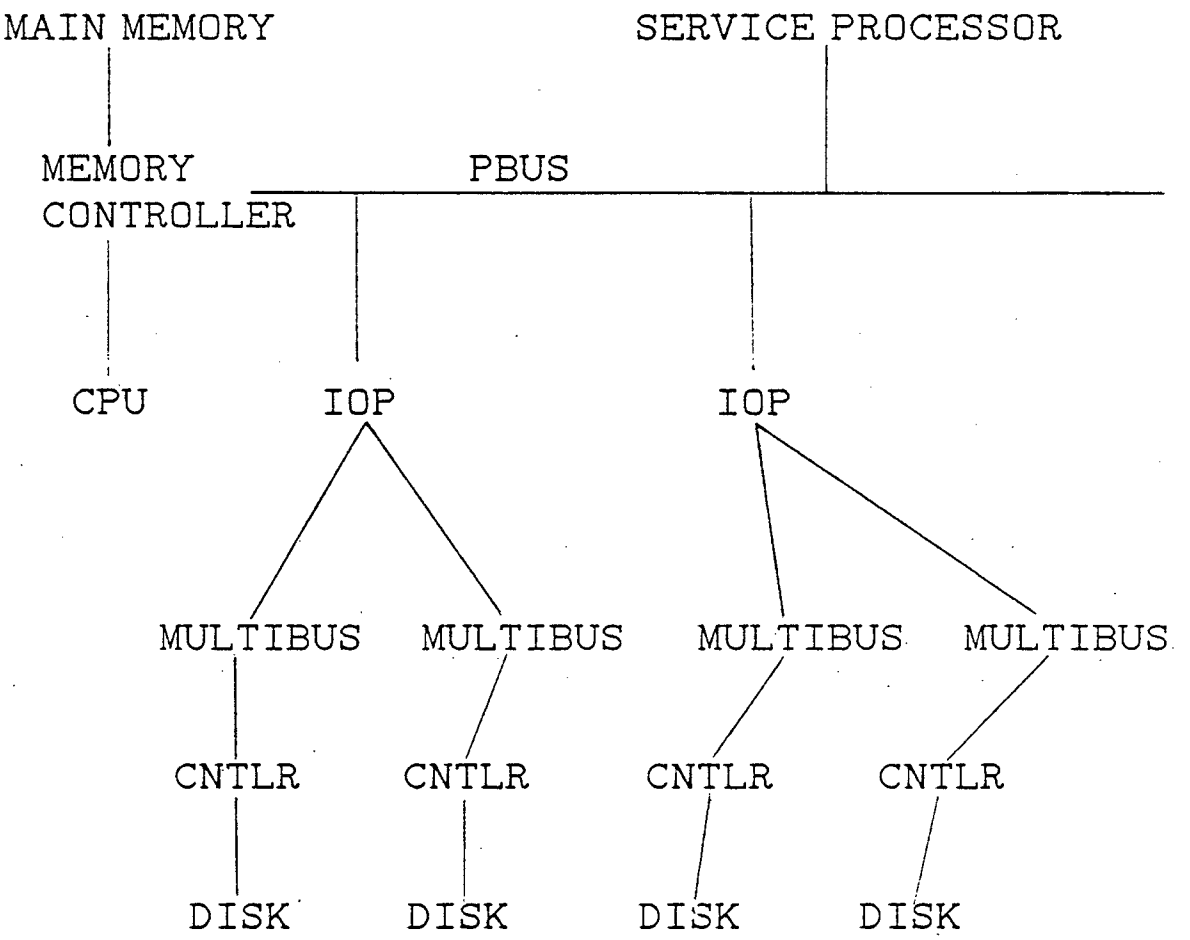
# DISK STRIPING

- Disk striping allows file systems to span partitions
  - o The partitions are usually not on the same drive
- Reasons for disk striping are:
  - o Create a file system larger than one drive
  - o Increase performance by balancing the disk load
  - o More flexibility in file system layout
- Striped file systems are made known to the kernel at boot time
  - o This is done by running */etc/putst* from */etc/rc*
- Striped file systems can be used like any other file system

# PERFORMANCE GUIDELINES FOR STRIPES

- Partitions should span two or more disks
- Put the most active file systems on VME disks (if possible)
- Partitions should span multiple controllers
  - o Partitions on the same controller cause sequential instead of parallel I/O operations
- Two portions of a stripe on one controller and the third on a second controller increases throughput *1 1/2* times
- Bandwidth limits performance
  - o Partitions should span Multi/VME busses

# SAMPLE CONFIGURATION



# DISADVANTAGES OF STRIPING

- Increased risk of data loss during hardware failure
- Large file systems make backup management more troublesome
- Decreased performance is possible with poorly chosen striping

# STRIPED FILE SYSTEM OVERVIEW

- Creating a striped file system requires the same steps as creating a new file system:
  - o Determine available partitions
  - o Shutdown to single-user
  - o Backup
  - o Create device files
  - o Umount file system(s)
  - o Modify */etc/fstab*
  - o Determine block/frag size
  - o Create the stripe(s)
  - o Make directory for the file system
  - o Run file system check on unmounted system
  - o Mount the file system
  - o Return to multi-user mode
  - o Restore as appropriate

# CREATING STRIPED FILE SYSTEMS

- To determine if there are existing stripes, use the */etc/getst* command
  - o Determine if there are the appropriate device files
  - o Create the device file(s) for the stripe if needed:

```
cd /dev
MAKEDEV stID#
```

- Striped file systems are created with the */etc/newst* command
- *newst* does the following:
  - o Creates a description of the stripe
  - o Loads the description into the kernel
  - o Creates a file system on the stripe partition
- *newst* format:

```
newst options raw_st_dev diskdev1 type1 diskdev2 type2 ...
```

- Some of the common *options* are:

| | |
|---|---|
| -b | *block_size* |
| -f | *fragment_size* |

# /etc/newst page 2

- *raw_st_dev* is the device name of the stripe file system
  - o */dev/rst*[*0-256*]
  - o *diskdev** is the name of the disk device to use
  - o *type** is the device type as named in */etc/disktab*
- Multiple device and type pairs can be specified

  Example:

```
# /etc/newst /dev/rst1 /dev/rda0d dkd-005 /dev/rda1d dkd-005
```

# THE /etc/stripecap FILE

- */etc/stripecap* contains descriptions of striped file systems
- *stripecap* entries are built by *newst* and are used by *putst* to build the kernel stripe table
- *stripecap* entries are in *termcap* format
- *stripecap* fields description:

| Name | Type | Description |
|---|---|---|
| np | num | number of partitions. |
| M | num | major device number of partition |
| m | num | minor device number of partition |
| D | num | number of partitions in section |
| B | num | interleave factor of section |
| S | num | number of blocks in section |

- Sample *stripecap* entry:

```
st2:\
  :np#2:\
  :M0#5:m0#36:\
  :M1#5:m1#9:\
  :D0#2:B0#170368:S0#8:\
  :D1#1:B1#85120:S1#8:
```

# SYSTEM-WIDE BOOT-TIME OPTIONS

- Allow optimization of performance
- Require no recompiling
- Specific to a system processor (CPU and CCU)
  - o CPU Central Processor Unit
  - o CCU Channel Control Unit
  - o MIOP Multibus I/O Processor
  - o HSP High Speed I/O Processor
  - o VIOP VMEbus I/O Processor
- */mnt/os/bootcmd.local* file on the SPU shows the processor configuration
- Sample entry from */mnt/os/bootcmd.local*

```
tune iop Accelerate_enable = 0
```

- View this file with the spu command:

```
spu -r /mnt/os/bootcmd.local
```

# TUNABLE PARAMETERS

- Parameters and their allowed values are listed in the SPU file */mnt/os/tunables* (DO NOT modify this file)
- Example set of tunable parameters:

| Parameter | Default | Minimum | Maximum | Processor |
|---|---|---|---|---|
| auto_nice_threshold | 600 | 1 | 9999999 | cpu |
| auto_nice_factor | 4 | 0 | 64 | cpu |
| dst_algorithm | 1 | 0 | 6 | cpu |
| enable_unique_core | 0 | 0 | 1 | cpu |
| erase_pattern | 0xAAAAAAAA | 0x80000000 | 0fFFFFFFF | cpu |
| erase_unlink | 0 | 0 | 1 | cpu |
| fp_default_mode_ieee | 0 | 0 | 1 | cpu |
| gateway | 0 | 0 | 1 | cpu |
| ipforwarding | 1 | 0 | 1 | cpu |
| ipsendredirects | 1 | 0 | 1 | cpu |
| logresume | 4 | 0 | 100 | cpu |
| logsuspend | 2 | 0 | 100 | cpu |
| maxusers | 32 | 8 | 256 | cpu |
| max_user_processes | 40 | 4 | 150 | cpu |
| nfs_portmon | 0 | 0 | 1 | cpu |
| nstbuf | 512 | 128 | 8192 | cpu |
| number_ptys | 64 | 0 | 256 | cpu |
| number_tty_controllers | 2 | 0 | 16 | cpu |
| stripe_devices | 16 | 4 | 256 | cpu |
| subnetsarelocal | 1 | 0 | 1 | cpu |
| ta_force_EOF_on_close | 1 | 0 | 1 | cpu |
| time_zone | 360 | -720 | 720 | cpu |
| udpcksum | 0 | 0 | 1 | cpu |
| accelerate_enable | 1 | 0 | 1 | iop |
| ca_timer_code | 8 | 0 | 15 | iop |
| uv_num_windows | 512 | 128 | 960 | viop |

- See Chapter 3 of System Managers Guide

# SETTING TUNABLE PARAMETERS

- Entries which modify system tunable parameters have the format:

  o tune *processor parameter* = *value*

- *Processor* is *cpu, hsp, iop, viop*

- Examples for */mnt/os/bootcmd.local* file on the *spu*:

```
tune cpu auto_nice_threshold = 100
tune cpu auto_nice_factor = 10
tune cpu max_user_processes = 100
tune cpu enable_unique_core = 1
tune cpu ta_force_eof_on_close = 0
```

- When device drivers are added if necessary modify */mnt/os/tunables*

- Additional swap space can be added up to 16 b partitions by modifying */etc/fstab*

- Non-b swap partitions are defined in */mnt/os/bootcmd.local* as follows:

```
swap on da1g
```

# FILE SYSTEM SPACE

- File system efficiency degrades when a partition becomes more than 90% full
  - o Only the superuser can allocate space when this happens
- Applications may not work correctly if the file system's allocated space is reaching its limit
  - o The disk quota system limits users space to avoid this problem
- Utilities are available to display the amount of free space in file systems
- Utilities also display space utilized by users

# FILE SYSTEM QUOTAS

- Quotas are used to restrict the amount of space that can be allocated to a user
- Quotas work on a per-user, per-file system basis
- Quotas do not have to be enabled for all users or all file systems
- There are two kinds of limits imposed by the quota system:
  - o The hard limit cannot be exceeded
  - o The soft limit will print a warning message whenever this limit is exceeded
  - o A time limit is imposed with soft limits
  - o If the allocated space does not drop before the timeout is reached the hard limit is enforced
- Quotas control allocated data blocks and i-nodes

# SETTING UP QUOTAS

- Setting up quotas for a file system requires the following steps:
  - o Create an empty file, *quotas*, in the root directory of the file system
  - o Initialize the file with the */usr/etc/quotacheck* program
- Create limits for users with the */usr/etc/edquota* utility
- Enable quotas with the */usr/etc/quotaon* utility
- Enable quotas automatically at boot time from */etc/fstab*
- Enter the following in the */etc/rc.local* file:

```
/usr/etc/quotacheck -a
/usr/etc/quotaon -a
```

# INITIALIZING THE *quotas* FILE

- The *quotas* file, located in the root directory of each file system with quotas, is initialized in two steps:
  - o Create an empty file *quotas* in the root directory of the file system

    ```
    touch /mnt/quotas
    ```

  - o Initialize the values in the *quotas* file by running */usr/etc/quotacheck* on the file system

- The *quotacheck* command format:

  `/usr/etc/quotacheck` *file_system*

- Do this on every *file system* that is to have quotas
  - o *quotacheck* can be executed using the *a* option to initalize all file systems listed in the */etc/fstab* file

# CREATING QUOTA LIMITS

- Quota limits for a user are created with the */usr/etc/edquota* command

  `/usr/etc/edquota` *user*

- The *vi* editor (or the editor specified in the EDITOR environment variable) is invoked on the user's quota records

- When the *-b, -B, -i, -I* batch options are used, no editor is invoked

  `edquota [-b` *bsft#* `-B` *bhrd#*
  `-i` *isft#* `-I` *ihrd#* `]` */fs_nm user(s)*

- To set a soft block limit of 500 for several users on the */mnt* file system (using the batch format)

  `edquota -b 500 /mnt smith jones larson`

- Existing quota limits can be copied from one user to another by using *edquota* with the *-p* option

  `edquota [-p]` *prot-user user(s)*

  Example:

  `edquota -p smith jay brown`

# ENABLING FILE SYSTEM QUOTAS

- Quotas are enabled with the */usr/etc/quotaon* command
- The *quotaon* command format:

  */usr/etc/quotaon /file_system*

- Quotas can be disabled with the */usr/etc/quotaoff* command
- The *quotaoff* command format:

  `/usr/etc/quotaoff` */file_system*

- Quotas are automatically disabled when *file_system* is unmounted
- If quotas have been disabled, *quotacheck* should be run before they are enabled again
  - o This will account for any activity that occurred while quotas were disabled

# AUTOMATING QUOTAS

- The *-a* option causes *fstab* to be read when running *quotacheck*, *quotaon* and *quotaoff* to determine which file systems should have quotas
  - o The file system's *fstab* entry should contains the option *quota*

- Example *fstab* entry for a file system with quotas:

```
/dev/da0d /mnt1 4.2 rw,quota 1 1
```

- To invoke quotas at boot time add these lines to the */etc/rc.local* file:

```
/usr/etc/quotacheck -a
/usr/etc/quotaon -a
```

# DISPLAYING CURRENT QUOTAS

- The *quota* command displays the user's current usage and limits
  - o Non-superusers can only check their own quotas:

```
% /usr/ucb/quota -v
```

- To check other user's quotas, enter:

```
# /usr/ucb/quota -v user
```

- Without the -v option, nothing will be displayed for file systems that are not over quota for that user

- To display file system usage and quotas, use:

```
# /usr/etc/repquota
```

- The *-a* option can be specified instead of *file system* to check all default quota file systems

# CHECKING FILE SYSTEM FREE SPACE

- The *df* command displays the amount of space used and available in file systems
  - o The space is listed in kilobytes
  - o The *df* command format:

    `df` [*file_system*]

- If no file system is specified, free space for all mounted file systems listed in */etc/fstab* are given

- The *-i* option displays the used and free i-node count

- The command:

  *df* . displays current file system

- The command:

  *df -i* displays blks and frags

# DETERMINING DISK USAGE

- The *du* command lists the amount of space taken by a directory and its contents
  - o Space is listed in kilobyte blocks
- The *du* command format:

  `du [`*options*`] [`*directories*`]`
- The *-s* option displays only the grand total for *directories*

  Example:

  `% du -s /mnt/smith`

# DISPLAYING FILE SYSTEM USAGE

- The */usr/etc/quot* command displays the amount of space taken by a user on a file system

- The *quot* command format:

  /usr/etc/quot *options file system*

  - o The -a option can be specified instead of *file system* to display values for all mounted file systems
  - o The -f displays number of files as well as the file space owned by each user
  - o The -v option displays totals for files not accessed in the last 30, 60, or 90 days
  - o Example:

```
# /usr/etc/quot /mnt
/dev/da0h (/mnt):
24530   johnson
 1693   jolster
 1115   smith
  642   conners
   43   anderson
    8   jones
    3   #153
    1   test
```

# THE *OP* UTILITY

- Allows execution of certain root operations without superuser privileges
  - o Dumps
  - o Tape handling
  - o System shutdown
  - o Killing processes
- Requires defining commands and who can execute them in the */etc/op.access* file
- Using the */etc/op.access* file
  - o Verifies that the operator can execute the command
  - o Validates any variable arguments
  - o Sets up specified attributes (uid, gid, umask, etc.)
- Logs attempted executions (successful and unsuccessful) via *syslogd*
- An operator is denied direct access to the */etc/op.access* file

# THE *OP* UTILITY (page 2)

- An operator can obtain help and a list of executable commands with the format:

```
/etc/op -h
```

- The *op* command requires an *access_name*
  - o command (defined in */etc/op.access*)

```
/etc/op access_name [access_arg ...]
/etc/op dumpmnt
```

- Each *access_name* can require either literal or variable arguments

```
/etc/op dump /
```

# *op.access* FILE

- Describes operator-type functions (access name and restrictions) to be used by the *op* utility
- Operator-type functions are defined by the system administrator
  - o No default file exists
- The attributes that must be defined:
  - o Access name--alphanumeric starting in column one
  - o Full pathname of executable--follows access name preceded with white space
  - o Arguments (variable args can be represented with $n) for the executable
  - o Specific settings or restrictions (uid, gid, dir, umask, groups, users)
- File is read only (0400) for security reasons

# The *op.access* SETUP FORMAT

- The file format:

  *access_name command_path* [*$1/arg,arg ...*]; [ *restrict=value*]

  ```
  dump5 /etc/dump 5Gu $1;
  $1=/,/project users=smith
  ```

- The form can be several lines; all continuation lines can begin with any character EXCEPT alphanumeric

- The *access_name* is a required alphanumeric field beginning in column 1
  - o (This is the parameter used with the *op* utility)

    ```
    shutdown
    ```

- The *command_path* is a required absolute pathname of the executable to run when the *op* command is invoked with the *access_name*

  ```
  shutdown /etc/shutdown
  ```

- The *args* are any number of literal arguments needed by the *command* followed by a semicolon

  ```
  shutdown /etc/shutdown -h now;
  ```

# The *op.access* SETUP FORMAT (page 2)

- *args* can be variables:
  - o Variable *args* are specified as $1 ... $n
  - o They have the form of $n=value
  - o Expressions are allowed as value
  - o The value can be single or multiple selections separated by commas or no value

```
#Note this is an
#incomplete example
#no restrictions provided
#
#A single variable selection
#$1 must test as now at execution
#$2 has no value; can be anything
shutdown /etc/shutdown -h $1 $2; $1=now
#
#multiple variable selections
#operator must enter variable
tape /etc/tpc $1 $2;
$1=enable,disable,start,stop,restart
$2=all,unit0,unit1
```

- Operator executes the command:
  /etc/op shutdown now "Disk is full"
  /etc/op tape start unit1

- The commands are executed:

```
/etc/shutdown -h now "disk is full"
/etc/tpc start unit1
```

# The *op.access* SETUP FORMAT (page 3)

- The restrictions are a set of optional parameters which specify settings for:
  - o uid--numeric user ID or login name
  - o gid--numeric group or group name
  - o dir--change directory to path specified
  - o chroot--change the root directory to the path specified
  - o umask--set the file creation umask to value specified
  - o groups--any user belonging to this group can execute the command
  - o users--allow only these users to execute the command

- The command:

```
dump5 /etc/dump 5Gu $1; $1=/,/project
  users=smith groups=opgrp
```

- Executed by user *smith* or any user belonging to the group *opgrp*
  - o To execute:

```
/etc/op dump5 /project
```

# The *op.access* SETUP FORMAT (page 4)

- To set default values for all entries, use:

  DEFAULT *spec=value*

  ```
  DEFAULT groups=opgrp
      users=smith,jones umask=077
  ```

  as the first non-comment line

- If an entry defines its own restrictions, the DEFAULT is overridden

  ```
  dump /etc/dump 0Gu $1;
  $1=/,/project
      groups=dumpers users=james
  ```

- Only members of the *dumpers* group or user *james* can execute the command

- Using `users=` or `groups=` sets the value to null

- For other attributes (uid, gid, dir, chroot, umask), a null value sets the attributes of the *op* program

## */usr/etc/syslog.conf*

- *syslogd* reads and logs messages into files described in */etc/syslog.conf*
- Entries in */etc/syslog.conf* determine types of messages to log and where to log them
- Each entry has a selector to determine the message priorities and an action
- The format of each entry:
  *facility.level*; *send_message_here*
- The *facility* and *level* are always separated by a dot and the entry ends with a semicolon
- The facilities available are:

| | |
|---|---|
| kern | messages generated by the kernel |
| user | messages generated by user processes |
| mail | messages generated by the mail system |
| daemon | messages generated by system daemons |
| auth | messages generated by authorization system |
| lpr | messages generated by line printer spooling system |
| local0-7 | reserved for local use |

- More than one facility may be delimited with a comma
  - o An asterisk indicates all facilities

# SETTING UP THE *syslog.conf* FILE

- The levels available are:

| | |
|---|---|
| emerg | panic condition; normally to all users |
| alert | condition that should be corrected immediately (corrupted database) |
| crit | critical conditions (hard-device errors) |
| err | errors |
| warning | warning messages |
| notice | conditions that are not error conditions but require attention |
| info | informational messages |
| debug | information of use for debugging a program |

- If a message cannot be passed to *syslogd*, it attempts to write the message on the console

- Format alternatives for where messages are to be logged:
  - o Absolute pathname
  - o Hostname preceded with @
  - o A comma separated list of users (receive messages if logged on)
  - o An asterisk (sends message to all logged in users)

# SETTING UP THE *syslog.conf* FILE (cont.)

- The default file can be modified:

```
*.err;kern.debug;auth.notice    /dev/console
*.err;kern.debug;\
daemon,auth.notice;mail.crit    /usr/adm/messages
lpr.debug                       /usr/adm/lpd-errs
mail.debug                      /usr/spool/mqueue/syslog
*.alert;kern.err;daemon.err     operator
*.alert                         root
*.emerg                         *
```

- An entry containing * must be at the beginning of the line

- After modifying the default file, kill and restart *syslogd*

- To start the appropriate daemons at boot-time, the */etc/rc.local* file must be modified:

```
if [ -f /usr/etc/syslogd ] ; then
rm -f /dev/log
/usr/etc/syslogd &
echo 'starting system logger' > /dev/console
fi
```

# The *logger* UTILITY

- Makes entries in the system log
- A message can be given on the command line and is logged immediately
- A message can be logged from a file (*-f*)
- Messages can be entered with a specified priority (*-p*) using the *facility.level* pair
- The format:

```
logger [options] [message]
logger -p mail.debug "sendmail
keeps dying"
logger -p mail.debug -f
/bin/message
```

# The *verify* UTILITY

- Aids in diagnosing system problems
- Uses database files in */usr/lib/verify* to verify characteristics for specified files - file type, owner, group, and mode
- Default database files (you can add your own):
  - o 68k
  - o nfs
  - o batch
  - o system
  - o net
- Directories are specified with `:dir:` *dirname*; file specifications follow each *dir* listing
- Each file specification contains colon separated fields of *file name*, *mode*, *owner*, and *group*

# THE *verify* UTILITY (cont.)

- Entries can specify the file type:

| | |
|---|---|
| b | Block special |
| c | Character special |
| d | Directory |
| l | Symbolic link |
| p | Named pipe |
| s | Socket |

- Example database file:

```
:dir:usr
convex:775:root:bin:t=d:
doc:775:root:bin:t=d:
etc:775:root:bin:t=d:
infosys:775:root:bin:t=d:
lib:775:root:bin:t=d:
spool:775:root:bin:t=d:
ucb:775:root:bin:t=d:
```

- If an asterisk appears in the first position of the fields *owner, group, mode*, then that field is not checked

- If a *?* mark is in the first position of the field *owner or group*, then the file may have any owner or group listed in the password and group file

# USING THE *verify* UTILITY

- To output a list of differences, use:

  `verify` *database_file*

  No changes are made to the files

- To interactively change *owner*, *group*, or *mode*, use:

  `verify -i` *database_file*

# THE */etc/cron* UTILITY

- */etc/cron* executes programs at specified dates and times
- *cron* is a daemon started from */etc/rc.std*
- Entries in *crontab* files determine which processes to run and when to run them
- *crontab* entries are in the file */usr/lib/crontab* and in *.crontab* files located in users' login directory
- *crontab* must be supplied absolute path names to */etc/cron*

# .CRONTAB FILES

- Users can place *crontab* entries in a file named *.crontab* in their home directory
- *cron* checks *.crontab* files for modification once an hour
- If *.crontab* has been modified, current *cron* commands for the user are replaced with the *.crontab* contents
- The *tellcron* utility causes *cron* to check *.crontab* immediately
- *tellcron* takes no parameters or options

# CRONTAB ENTRIES

- *crontab* entries have the following format:

  *min hr day_of_mnth mnth_of_yr day_of_wk prgrm*

- The first 5 fields each contain an integer value

- Multiple values for a specific field can be specified the following ways:
  - o Lists of comma-separated values
  - o Ranges of '-' separated values
  - o The asterisk specifies all legal values

- Example *crontab* entries:

```
30 * * * * /usr/adm/accounting > /dev/console
40 04 * * * find / -name '.#*' -atime +3 -exec rm -f {} \
01 11 * * 7 /bin/sh /usr/spool/uucp/shorten
37 01  *  *  * /bin/sh /usr/lib/diskspace
00 08,12,16 * * 1-5 /bin/sh /usr/lib/users
```

# PRIORITIES

- Each process has a priority in the range of -64 to +64
  - o The priority helps the process scheduler determine which process will be run
- Lower priority numbers mean higher priority
- User processes run with priorities in the range of -32 to +32 with 0 as the default
- The *nice* utility invokes processes with a non-default priority
- The *nice* command format:
  `nice` *number command*
- *number* is the new priority
  - o Number must be preceded by + for positive priorities, or a - for negative priorities
  - o Only the superuser can specify negative priority values

  Example:

```
# nice -20 ps aux | less
# nice +64 sleep 1000 &
```

# THE */etc/renice* COMMAND

- The *renice* command changes the priority of running processes
- Only the superuser can lower nice values
- The format of the *renice* command is:

  `/etc/renice` *number processes*

- *processes* can be a combination of:
  - *-p* followed by process IDs.
  - *-u* followed by a list of users.
  - *-g* followed by a list of process groups.
- *number* is the new priority to be assigned to the processes

Example:

```
# /etc/renice +20 -u smith
```

# MONITORING SYSTEM ACTIVITY

- There are several utilities available to monitor system activity
- These utilities can be used to determine:
  - o The amount and kind of activity occurring
  - o The system resources being used
  - o The resource efficiency

# THE ps UTILITY

- The *ps* utility describes the status of running processes

- By default, *ps* lists processes with the same UID as the user running *ps*

  o This can be changed with the options:

| | |
|---|---|
| a | All processes associated with terminal |
| g | All processes for the current user |
| t*tty* | Processes belonging to process group |
| x | Processes not assigned to a terminal |
| #*pid* | List only this process |

Example:

```
% ps -t03
```

# THE *ps* COMMAND OUTPUT FIELDS

- The following output fields can be produced by *ps*:

| Title | Description |
|---|---|
| PID | Process ID. |
| TT | Control terminal. |
| TIME | CPU time. |
| STAT | Process state. |
| COMMAND | Command. |
| USER | User name. |
| %CPU | CPU utilization. |
| NI | Nice value. |
| RSS | Resident set size. |
| SIZE | Virtual size. |
| LIM | Soft memory limit. |
| TSIZ | Text image size. |
| TRS | Resident set size of text image |
| RE | Incore time. |
| SL | Sleep time. |
| PAGEIN | Page requests requiring disk I/ |
| UID | User ID. |
| PPID | Parent process ID. |
| CP | Short term CPU utilization. |
| PRI | Process priority. |
| WCHAN | Sleep address. |
| F | Process flags. |

# *ps* OUTPUT FIELDS (page 2)

- The *PID*, *TT*, *TIME*, *STAT*, and *COMMAND* fields are always listed .
- The following options define which additional fields are displayed:
- u (user oriented):

  USER %CPU %MEM SZ RSS

- v (virtual memory statistics):

  *SL RE PAGEIN SIZE RSS LIM TSIZ TRS %CPU %MEM*

- l (long listing)

  *F PPID CP PRI NI SIZE RSS WCHAN*

- Example:

*ps -u*

# THE *syspic* UTILITY

- The *syspic* utility displays the current system activity
- It can be used to determine if there are any problem areas in the system
- The format of *syspic* is:
  `syspic -p` *picture*
- *picture* can be one of the following:
  - o *system* is the default picture selected if *-p* is not used
  - o *disk* gives a picture of the disk activity
  - o *tty* gives a picture of the activity on TTY lines
  - o *network* gives a picture of the network activity
  - o *kluster* gives statistics on various size distributions from disk transfers and pageins

# THE *syspic* UTILITY (page 2)

- o *proc* gives information on processes
  - * Memory
  - * Per-CPU usage
  - * Top ten processes sorted by CPU usage
- o *vm* gives an overview of virtual memory

- The screen is updated every 5 seconds by default until the process is interrupted

Example:

*% syspic -p disk*

# CXBATCH SYSTEM OVERVIEW

- The batch system allows processes to be queued for execution
- There are advantages to using the batch system for long, time-consuming tasks:
  - o A CPU will not be overloaded with several of these processes running at one time
  - o Tasks can be assigned a higher or lower priority automatically
  - o Resources can be defined on a queue-by-queue basis
- Processes are placed on the batch queue voluntarily
- A queue daemon, */usr/lib/nqs/nqsdaemon*, processes all local queue requests
- The *netdaemon* processes remote transactions
- The *shepherd* daemon watches over batch jobs

# BATCH CONFIGURATION UTILITIES

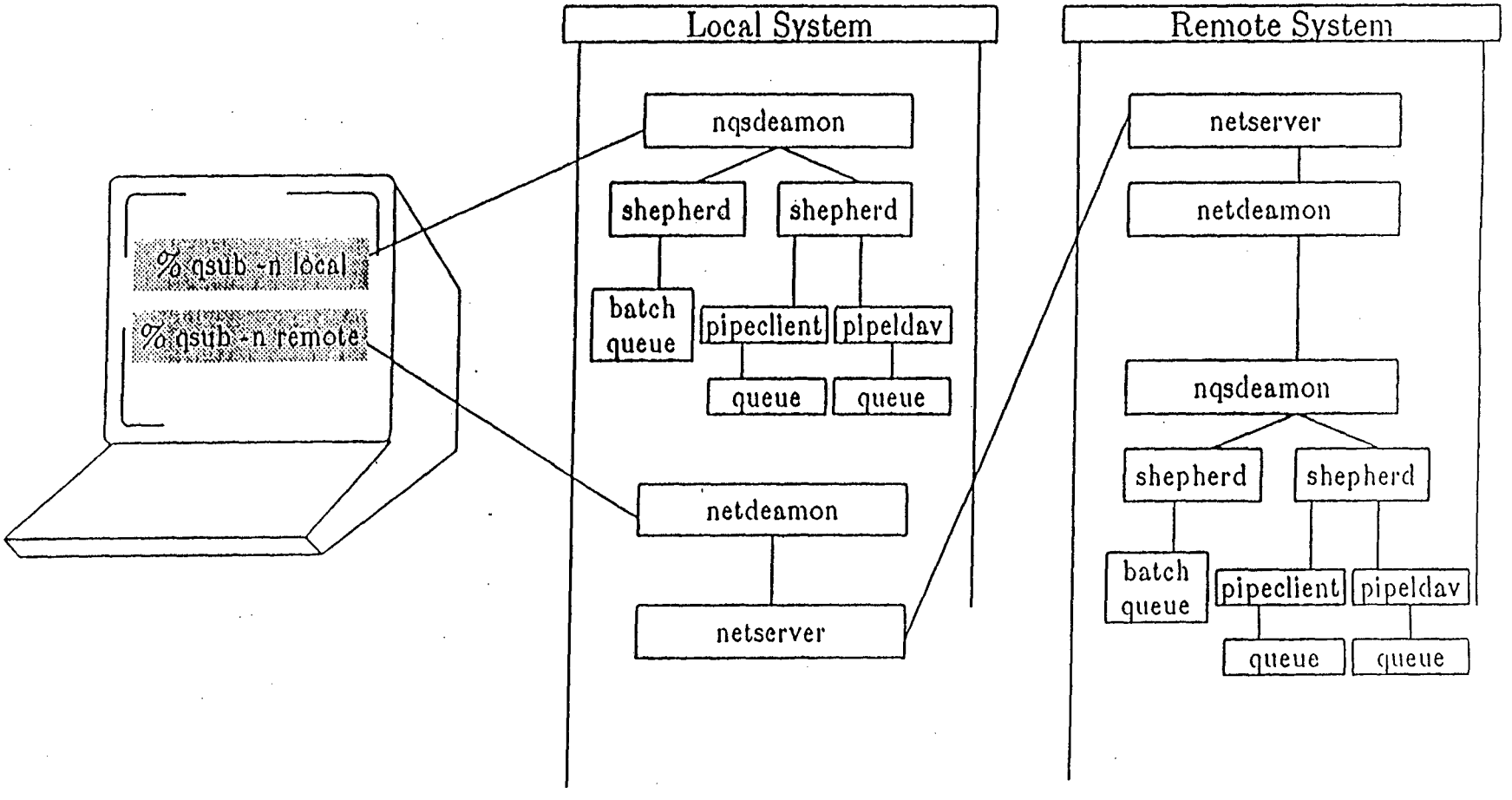
- There are two utilities for establishing and operating CXbatch:

| | |
|---|---|
| *qmapmgr* | *qmapmgr* builds and maintains the network database; it establishes a mapping between CXbatch and each machine in the CXbatch configuration. |
| *qmgr* | *qmgr* controls requests, queues, and general CXbatch configuration on a local system. |

- Two types of CXbatch queues to route and execute batch requests:

| | |
|---|---|
| *batch* | only executes batch requests; does no routing |
| *pipe* | transmits requests to other batch queues locally or remotely |

# FLOW OF CXBATCH REQUESTS



# SETUP OVERVIEW

- Setting up batch queues requires the following:
  - o To allow submitters to import/export working directories NFS must be running
  - o Verify that the daemon programs, *nqsdaemon*, *netdaemon*, and *logdaemon* are running and and that an entry for *nqsdaemon* appears in the *rc.local* file
  - o Create the CXbatch configuration database with the *qmapmgr* command
  - o Determine type (batch or pipe) of queue to be configured on each system
  - o Create queues and define attributes of each queue on each system with *qmgr* utility
  - o Edit */etc/hosts.equiv* file to allow access to remote queues or notify users to create a *.rhosts* file
  - o Create accounts for users who do not have accounts on the CXbatch machines

# SETUP OVERVIEW (page 2)

- Continuation of setting up batch queues
  - o Notify users that *.login*, *.cshrc*, *.logout*, and *.profile* files may need to be modified
  - o Notify users of commands and options available with CXbatch

# STARTING CXBATCH DAEMONS

- The following daemon programs must be running to install, create, or modify CXbatch queues:

  *nqsdaemon* — Handles all local transactions, including submits, deletes, job scheduling, and system configuration

  *netdaemon* — Handles all remote transactions

- The *logdaemon* must be running to record error messages
  - o *nqsdaemon* and *netdaemon* contact *logdaemon* when they need to print an error message

- These three daemons are started from */etc/rc.local* with the entry:

```
if [ -f /usr/lib/nqs/nqsdaemon ]; then
  /usr/lib/nqs/nqsdaemon > /dev/console
  /bin/echo "Started CXbatch daemons." > /dev/console
fi
```

- During the CXbatch process, *netdaemon* may exec a *netserver*
  - o *nqsdaemon* will fork a *shepherd* to run/route the job

# THE *qmapmgr* UTILITY

- Use *qmapmgr* to build the network database

- The network database contains:

| | |
|---|---|
| mid | The machine identification number used by CXbatch to identify a specific machine |
| hostname | This serves as the principle name and must be contained in the */etc/hosts* file |
| alias | Names, other than hostname, of machines in the network |
| uid | User ID number; a user ID on one machine can be mapped to a user ID on the local machine |
| gid | Group ID number; a group ID on one machine can be mapped to a group ID on the local machine |

The format:

```
/usr/convex/qmapmgr MAPMGR:
keyword parameter
```

# INITIAL CONFIGURATION

- Create the initial database for CXbatch (installation procedure) with the *qmapmgr* utility

- The format:

  `/usr/convex/qmapmgr`

  MAPMGR: `create`

- During installation assign machine identification numbers (MID) for all machines running CXbatch with the *qmapmgr* utility

- The format:

  MAPMGR: `add mid 1` *hostname*

- To view a list of machines MIDs, use:

  MAPMGR: `show`

- Information can be added or deleted from the database any time with *qmapmgr*

- A default configuration may be selected during installation

# THE *qmgr* UTILITY

- The *qmgr* utility provides a variety of control and configuration capabilities:

  - o abort
  - o add
  - o add/change attributes
  - o create
  - o delete
  - o enable
  - o disable
  - o move
  - o start
  - o stop

- With the *qmgr* utility, operators and managers can control and configure queues

# CXBATCH MANAGERS AND OPERATORS

- The *qmgr* utility allows users to be identified as either *manager* or *operator* of CXbatch

| | |
|---|---|
| manager | A user who may change any CXbatch characteristics on the local machine |
| operator | A user who may execute only the operator commands which are a subset of the *qmgr* commands (A list of operator commands is provided in the *CONVEX CXbatch System Manager's Guide*) |

The *op* utility can be used to allow access to specific *qmgr* commands

- Users have access to the following *qmgr* commands:
  - o exit
  - o help
  - o hold
  - o show
  - o move
  - o release
  - o request

# MANAGER AND OPERATOR PRIVILEGES

- To allow a user manager privileges, use the *m* option:

  /usr/convex/qmgr

  Mgr: add manager *username*:m

  Mgr: add manager *userid@hostname*:m

- To allow operator privileges, use the *o* option:

  /usr/convex/qmgr

  Mgr: add manager *username:o*

  Mgr: add manager *userid@hostname:o*

- To display a list of authorized CXbatch managers and operators, use the *show managers* command:

  /usr/convex/qmgr

  Mgr: show managers

# DEFAULT QUEUES

- To select a default (local) configuration when installing CXBATCH:
  - o short queue
  - o long queue
  - o verylong queue
- Additional queues (batch or pipe) can be defined specifically for your system(s)
- The number of queues does not indicate the number of jobs that can be run at one time
  - o One or more jobs can be executed from each queue
  - o Jobs can be assigned to queues based on system activity
- Resource limits may be set for each queue

# CUSTOMIZING THE CXBATCH SYSTEM

- Match workload to available resources
  - o Single-machine system
  - o Multiple machines
  - o Site-specific situations
    - * CPU time required
    - * Machine load
    - * Machine use
- Batch queues vs. pipe queues:

| | |
|---|---|
| Batch | routes to specific queues |
| Pipe client | routes to first queue in its destination list that is available with required resources |
| Pipeldav | routes to destination queue with the lowest load factor that can accept the job |

# CONFIGURING A BATCH QUEUE

- Use the *qmgr* utility to create and customize batch queues

  /usr/convex/qmgr

  Mgr: create batch_queue *queuename* pr=*queuenumber*

- The *pr* option is used to determine servicing (inter-queue priority) of queues

- Enable the queue(s) to allow them to accept jobs:

  Mgr: enable queue *queue_name*

- Start the queue(s) to allow requests to run:

  Mgr: start queue *queue_name*

- To allow queues to run requests during specific hours, add an entry to the */usr/lib/crontab* file indicating time to start and to stop each queue:

  0 18 * * * /usr/convex/qmgr start queue *queuename*

  0 7 * * * /usr/convex/qmgr stop queue *queuename*

- Any requests in the queue that are running when the *stop* command is executed are allowed to complete

# SETTING QUEUE ATTRIBUTES

- Specific resource limits and/or attributes can be set for each queue

- Default resources/attributes include:

| ATTRIBUTE | DESCRIPTION |
|---|---|
| | |
| run_limit | Determines the maximum number of requests that can be run simultaneously. |
| accounting | Turn accounting on/off for CXbatch. |
| activity_id_offset | A number that is added to a job's activity ID for maintaining accounting information. |
| no_access | Used in conjunction with *add users/groups* parameter; allows access to be restricted to specific users or groups. |
| import directory | Determines whether submitter's current directory will be imported or the submitter will be allowed to import the directory. To allow importing of directories, NFS must be running. |
| corefile_limit | Determines maximum size of core file. |
| data_limit | Determines maximum size in bytes of the data segment for a process. |
| per_process permfile_limit | Limits maximum size of a file created by any process within the request. |

# QUEUE ATTRIBUTES (page 2)

| ATTRIBUTE | DESCRIPTION |
|---|---|
| | |
| nice_value_limit | Determines the proportion of CPU time allocated to a process relative to all other processes in the system. The lower the value, the higher the proportion of CPU time allocated to that process. |
| stack_limit | Maximum size allowed for a process. |
| per_process cpu_limit | Limit placed on per-process maximum CPU time limit; if time is exceeded, process is killed by default. |
| working_set_limit | Limit on physical memory. |

- Users can specify limits that are applied to each process request
  - o CXbatch compares the limits specified for each request with established limits
  - o Requests exceeding limits are rejected

# DEFAULT QUEUE VALUES

- Default attributes assigned to the queue:

  Run_limit = 1
  Accounting: Off
  Activity ID offset: 0
  Unrestricted access
  Import directory: Not available
  Per-process core file size limit = UNLIMITED
  Per-process data size limit = UNLIMITED
  Per-process permanent file size limit = UNLIMITED
  Per-process execution nice value = 0
  Per-process stack size limit = UNLIMITED
  Per-process CPU time limit = 36000.0
  Per-process working set limit = 10 megabytes

- Change resource limits on a host machine with the *qlimit* command:

  `/usr/convex/qlimit` *[hostname]*

- The format for setting the resource limits:

  Mgr: `set` *attribute=(value) queuename*

  Mgr: `set` *attribute queuename*

# SETTING QUEUE VALUES

- To set the run limit:

  Mgr: `set run_limit=2` *queuename*

- To set the CPU limit:

  Mgr: `set per_process cpu_limit=(5:00)` \
  *queuename*

- To limit access to specific users:

  Mgr: `set no_access` *queuename*

  Mgr: `add users=(smith, jones)`
  *queuename*

- The *qmgr(8)* man page provides a complete listing of all parameters and the format for each attribute

# DISPLAYING QUEUE ATTRIBUTES

- To verify queue attributes, use the *show* command with the *qmgr* utility:

  Mgr: show *attribute*

| ATTRIBUTE | DESCRIPTION |
|---|---|
| all | Displays limits supported, managers, parameters, and queues. |
| limits_supported | Displays resource limit types which are meaningful on the local machine. |
| long queue | Displays long format of the status of all queues; supplies information on resource limits for each queue. |
| parameters | Displays the defaults for all queues on the local machine. |
| queue | Displays status information of all queues on the local machine. |

## *pipeclient*

- The pipe client is a program that decides destination for queue requests
  - o Based on queue resource limits
  - o Based on load average
- Batch requests are forwarded through the pipe queue to the destination batch queue
- A server must be identified when the queue is created
  - o *pipeclient* - routes to the first destination queue that meets resource requirements
  - o *pipeldav* - routes to destination queue with lowest load factor that meets resource requirements
- The pipe queue can route to the local or remote machine

# SETTING UP THE *pipeclient*

- Format to configure the *pipeclient* queue:

  /usr/convex/qmgr

  Mgr: create pipe *queuename* pr=# \
  server=(/usr/lib/nqs/pipeclient) \
  dest=(*queuename@host,queuename@host*)

- Commands can be entered on more than one line
  - o Continuation line must end with a space and a slash (\)

- To set up the *fc* queue to route to *local* queue *fast* and *remote* queue *best*:

  Mgr: create pipe fc pr=20 \
  server=(/usr/lib/nqs/pipeclient) \
  dest=(fast@local,best@remote)

- *enable* and *start* the queue and set appropriate resource limits

# LOAD BALANCING

- The routing of a job can be done using the principle of load balancing
- Load balancing places job based solely on a modified load average algorithm:

$$mod\ ld\ avg = (ld\ avg + (q\ lngth * weight))/scale$$

- The *pipeldav* program maintains the load average information on a queue-by-queue basis
  - o *pipeldav* reads the load information for each host
  - o Contacts the *netdaemon* to get queue information
  - o Increments the load average by a weighting factor for each job in the queue
  - o Gives the job to the best processor
  - o weight - is given per queue
  - o scale - is given per machine
  - o C240 gets higher scale
  - o C1 gets lower scale

# SETTING UP THE *pipeldav*

- The general format to configure the *pipeldav* queue:

```
/usr/convex/qmgr
Mgr: create pipe queuename pr=# \
server=(/usr/lib/nqs/pipeldav [-w#] [host scale]) \
dest=(queuename@host,queuename@host)
```

- To set up the *vc* queue to route to *local* queue *fastest* and *remote* queue *c* with a weight of **2.0**:

```
Mgr: create pipe fc pr=20 \
server=(/usr/lib/nqs/pipeldav -w 2.0) \
dest=(fastest@local,c@remote)
```

- The *w* option is used to set a weighting factor greater than the default (**1.0**)

- Different scale values can be assigned to each destination queue:

```
server=(/usr/lib/nqs/pipeldav local 2.0 \
    remote 3.0)
```

- *enable* and *start* the queue

# MODIFYING FILES FOR REMOTE ACCESS

- Users must have an account on each system where their jobs will be executed
- For CXbatch execution, each user must have access to the hosts without password validation
  - o This may be accomplished in two ways:
  - o Create a system-wide file (*/etc/hosts.equiv*) of accessible hosts on each of the CXbatch machines
  - o Each user can create a file (*~/.rhosts*) containing a list of hosts
  - o Hosts are accessed without password validation on each system executing batch requests

# IMPORTING DIRECTORIES

- The import attribute on each queue:
  - o Determines how the queue gains access to the input files
  - o Determines access to files in the submitter's current working directory

- Set import attributes with `set import_dir` with the *qmgr* utility:

| ATTRIBUTE | DESCRIPTION |
|---|---|
| yes | Imports the current working directly unless explicitly restricted by the submitter |
| available | Imports the current working directory at request of submitter |
| no | Does not import the current working directory; rejects jobs that require input files to be imported |

- Files systems that are eligible for remote mounting must be listed in the */etc/exports* file

# ESTABLISHING A SHELL STRATEGY

- The shell strategy determines the interpretation of the batch commands
- Use the command `set shell_strategy` with the *qmgr* utility:

| ATTRIBUTE | DESCRIPTION |
|---|---|
| fixed | Specifies specifically which *shell* should be used to execute batch request scripts |
| free | Attempts to duplicate the *shell* choice that would have been made if the batch request script had been executed interactively |
| login | Specifies that the user's *login shell* be used to execute the batch request |

# MODIFY USER FILES

- CXbatch reads the user's login shell when jobs are executed
- Interactive commands in the *.login, .cshrc, .profile,* or *.logout* files will be executed during a batch job
- To prevent interactive commands from running during batch jobs, add the following test to the *.cshrc* file

```
if (! $?ENVIRONMENT) then
        . . .
        interactive commands
endif
```

- Another format:

```
if ($?ENVIRONMENT) exit
```

# BATCH REQUESTS

- Submit jobs to the batch queue system with the *qsub* command
- Submit batch requests with interactive commands or a script file
- When the job is complete, notification can be mailed to the user
- Files created during batch execution are written in the current directory on the local machine unless directed elsewhere
- Batch output is sent to *STDIN.o* and error messages are sent to *STDIN.e* in the current working directory
- When the *STDIN.o* file is created, it will contain the message
  - o *Warning: no access to tty; thus no job control in this shell; logout*
  - o To eliminate this message, the submitter must redirect output to another file

# CXBATCH SUBMITTER OPTIONS

- Options can be specified to control the environment in which the request is submitted and executed:
  - o Time to run the request
  - o Import or no import of current directory
  - o Intra-queue priority
  - o *nice* value
  - o Resource requirements
  - o Mail notification
  - o queue
  - o redirect standard output
- Options can be entered on the command line with the *qsub* command
- Options can be included in a script file submitted as a batch request
- The man page for *qsub(1)* contains a complete listing and description of the options

# SUBMITTING BATCH REQUESTS

- *qsub* has the format:

  qsub [*options*] *cmdfile*

  **or**

  qsub [*options*]
  *commands*
  ^D

- *cmdfile* is the name of the file containing the commands to run

- Commands are read from standard input if *file* is not listed

- Some of the options available:

| | |
|---|---|
| -a | Time to run request |
| -i | Import current directory |
| p | Set intra-queue priority |
| -q | Specify queue to which request is submitted |
| -me | Send mail when job has completed |

# SUBMITTING BATCH REQUESTS (page 2)

- Sending mail to the submitter upon completion using the *slow* queue:

```
qsub -me -q slow
find / -name "*.c" -user smith -print > findfile
^D
```

- Jobs can be submitted with a script:

```
#
# Example C shell batch request script
# Use @$ immediately preceding the options
#
# Send the user mail upon completion
# Submit the job to the slow queue
#
# @$-me -q slow
#
# To run the script at 11:00 this evening:
#
# @$-a "11:00pm CDT"
find / -name "*.c" -user smith -print > findfile
# The shell stops looking for imbedded commands
# the first command line in column 1
#
```

- Submit the job:

```
qsub myjob
```

- A copy of the script is kept in the file */usr/spool/nqs/scripts*

- Use *qjlist* to examine a job's script

```
qjlist request_ID
```

# DISPLAYING QUEUE INFORMATION

- *qstat* command examines the queue area and displays the following information:
  - o queuename
  - o queue type
  - o queue status
  - o number of requests in the queue
  - o request name
  - o request ID
  - o request owner
  - o intra-queue priority
  - o current request state

- Some of the available options are:

| | |
|---|---|
| -x *queue* | Lists the contents of *queue* and the resource limits |
| -a | Shows all requests; by default, only the requests for the user are shown |
| -l | Shows long form of information for a queued request |
| -m | Displays the date and time if the request was submitted with the *-a* option |

# USING THE *qstat* COMMAND

- The *qstat* command format is:

  `qstat` *[options] [queuename]*

  To display the status of the *short* queue:

  ```
  qstat short
  ```

```
short@convext; type=BATCH; [ENABLED, INACTIVE]; pri=30
 0 exit;  0 run;  0 stage;  0 queued;  0 wait;  0 hold;  0 arrive;
   REQUEST NAME   REQUEST ID  USER   PRI   STATE    PGRP
<2 requests RUNNING>
 3:   STDIN              127.convext  snoopy   24  QUEUED
```

- Only the submitter's jobs are shown

# USING THE *qstat* COMMAND (cont.)

- To display the resources available for a queue:

```
qstat -x short
```

```
short@convext; type=BATCH; [ENABLED, INACTIVE]; pri=30
 0 exit;  0 run;  0 stage;  0 queued;  0 wait;  0 hold;  0 arrive;
 Run_limit = 2;
 Accounting: On
 Activity ID offset: 0
 Cumulative system space time = 48.890000 seconds
 Cumulative user space time = 16.240000 seconds
 Unrestricted access
 Import directory: Yes
 Per-process core file size limit = UNLIMITED <DEFAULT>
 Per-process data size limit = UNLIMITED <DEFAULT>
 Per-process permanent file size limit = UNLIMITED <DEFAULT>
 Per-process execution nice value = 0 <DEFAULT>
 Per-process stack size limit = UNLIMITED <DEFAULT>
 Per-process CPU time limit = 36000.0 <DEFAULT>
 Per-process working set limit = 10 megabytes <DEFAULT>

 REQUEST NAME  REQUEST ID  USER  PRI  STATE  PGRP
1:   STDIN      128.convext      snoopy  24 RUNNING 28518
```

- To display all requests in a queue:

```
qstat -a short
```

```
short@convext; type=BATCH; [ENABLED, RUNNING]; pri=30
0 exit;  2 run;  0 stage;  1 queued;  0 wait;  0 hold;  0 arrive;

 REQUEST NAME  REQUEST ID  USER   PRI  STATE    PGRP
1:   STDIN      125.convext     mentor 24  RUNNING  28153
2:   STDIN      126.convext     mentor 24  RUNNING  28160
3:   STDIN      127.convext     snoopy 24  QUEUED
```

# REMOVING QUEUED JOBS

- Jobs can be removed from the queue with the *qdel* command
- The *qdel* command format for a submitter to remove his/her own request:

  `# qdel` *request_ID*

  `qdel 127`

- Only priviledged users can *remove* jobs submitted by others
  - o Superuser
  - o Batch operator
  - o Batch manager
- To remove one of the jobs belonging to user *snoopy*:

  `# qdel -u snoopy 126`

- To kill executing jobs, use the *-k* option

  `# qdel -k` *request_ID*

- To kill the process leader, the submitter may use:

  `# kill -1` *PGRP* or `kill -HUP` *-PGRP*

# MOVING JOBS

- Jobs can be moved from one queue to another with the *qmgr move my_request* command
- The job cannot be moved if the request exceeds queue limits or access restrictions of the new queue
- Batch operators and managers can move any job, regardless of restrictions
- The format for the *move* command:

```
/usr/convex/qmgr move my_request \
  request_id queuename

/usr/convex/qmgr move my_request \
    127 long
```

# HOLDING BATCH REQUESTS

- Submitters can prevent their queued jobs from starting execution with the *qmgr hold request* command
- Batch operators and managers can hold any submitter's job
  - o Only operators or managers can release the job
- The format to hold a job:

  ```
  /usr/convex/qmgr hold request request_id
  ```

  ```
  /usr/convex/qmgr hold request 127
  ```

- To release the job:

  ```
  /usr/convex/qmgr release request \
  request_id
  ```

  ```
  /usr/convex/qmgr release request \
  127
  ```

# THE PRINT SPOOLER

- The print spooler has the following features:
  - o Multiple printers
  - o Local or remote printers
  - o Multiple spooling queues
  - o Filters
    - * Text is sent through filters before being written to the print device
    - * Several sets of filters can be available for each printer
    - * The user printing the text can specify which filter to use
- Facilities to maintain and monitor the spool queues

# SETTING UP THE PRINT SPOOLER

- Define the print spoolers in the */etc/printcap* file
- Create the spool directory in */usr/spool*
- Create any non-standard filter programs
- Start the printer daemon (*/usr/lib/lpd*)
- Enable the queueing and printing of print requests with */etc/lpc*
- Create a list of trusted hosts in the */etc/hosts.equiv.pr* file

# THE */etc/printcap* FILE

- */etc/printcap* contains printer descriptions
- Each entry describes a printer in *termcap* format:

Some available fields are:

| Name | Type | Description |
|---|---|---|
| af | str | Name of accounting file |
| br | num | Baud rate if `lp` is a tty |
| fo | bool | Print a form feed when device is opened |
| if | str | Name of the accounting text filter |
| lf | str | Error logging filename |
| lp | str | Device name to open for output |
| of | str | Name of output filtering program |
| pl | num | Page length (in lines) |
| pw | num | Page width (in characters) |
| px | num | Page width in pixels (horizontal) |
| py | num | Page length in pixels (vertical) |
| rm | str | Machine name for remote printer |
| rp | str | Remote printer name argument |
| rs | bool | Restrict remote users to local accounts |
| sb | bool | Short banner (one line only) |
| sd | str | Spool directory |
| sh | bool | Suppress printing of banner page |
| tc | str | Entry of similar printer - must be last |
| tr | str | String to print when queue empties |

# /etc/printcap (Continued)

- Sample *printcap* file:

```
lp|rm|local line printer:\
    :af=/usr/adm/lpd-acct:if=/usr/lib/lpf:\
    :lp=/dev/lp0:sd=/usr/spool/lpd:\
    :lf=/usr/adm/lpd-errs:pu:mx#0:
lp1|line printer w/o filter:\
    :lp=/dev/lp0:sd=/usr/spool/lpd:\
    :lf=/usr/adm/lpd-errs:pu:
vaxip|long|rip|imagen-rip|remote imagen on the C-1:\
    :lp=:rm=convex:rp=ip:sd=/usr/spool/ripd:\
    :lf=/usr/adm/ipd-errs:
rm1|remote on p2:\
    :lp=:rm=toto2:rp=lp:sd=/usr/spool/rlpd:\
    :lf=/usr/adm/lpd-errs:pu:
imaprint|generic Imagen printer:\
    :pl=60:pw=80:px=2016:py=2624:\
    :sb:scS:sh:
imaspp|generic Imagen serial printer:\
    :br#9600:cf=/usr/local/lib/icif.s:\
    :df=/usr/local/lib/idvi.s:\
    :tc=imaprint:
```

# PRINT FILTERS

- The spooler passes the text to be printed through a spooler program
- Filter programs are designed to handle different printer types
  - o Some of the tasks they perform are:
    - * Initialize the printer
    - * Send the text
    - * Process special characters
    - * Process the end of printing text

# PRINT FILTERS (page 2)

- The standard output filter is specified in the `of` field in *printcap*
  - o Specialized filters can be defined in *printcap*
- The spooler invokes the filter
- The text to be printed is sent to standard input on the filter
- The filter's standard output is set to the printer device
- Page size parameters are passed to the `of` filter:

  `-w`*page_width*
  `-l`*page_length*

# THE SPOOLER FILES

- The spool directory is specified in the `sd` field in *printcap*
- There should be a separate spool directory for each printer
- Spool directories are subdirectories of */usr/spool*:

```
mkdir /usr/spool/printer_name
```

- Spool directories should be writable only by the user and group *daemon* (UID and GID 1)

```
# chall -o daemon -g daemon -m 770
spool_dir
```

# THE */usr/lib/lpd* UTILITY

- */usr/lib/lpd* is the printer daemon
- It is invoked from */etc/rc*
- When *lpd* is invoked, it restarts any jobs that were not finished when the system went down
- *lpd* will fork to handle print requests
- There may be more than 1 copy of *lpd* running at one time

# THE */etc/lpc* UTILITY

- */etc/lpc* is the printer control program
- Enables and disables printers and spooling queues
- Re-orders print jobs
- Displays printer status
- *lpc* commands include:

| | |
|---|---|
| `help` *command* | Help for *command* |
| `clean` *printer* | Clean *printer*'s queue directory |
| `enable` *printer* | Enable spool requests on *printer* |
| `start` *printer* | Enable printing on *printer* |
| `disable` *printer* | Stop *printer* from accepting requests |
| `stop` *printer* | Stop printing on *printer* |
| `status` *printer* | Display *printer*'s status |
| *printer job#* | Place a job at the top of queue |
| `exit` | Exit the *lpc* command |

# THE *lpr* UTILITY

- *lpr* utility initializes print requests
- *lpr* Specifies information to the spooler:
  - o The text to print
  - o The printer to use
  - o The filter to use
- The *lpr* command format:

  `lpr` *options files*

- *options* can be specified

| | |
|---|---|
| `-P`*printer* | Sends output to *printer* |
| `-m` | Mails a message to the user when printing is complete |

- *lpr* reads standard input if no files are specified
- Example:

```
% lpr data
```

# THE *lpq* UTILITY

- The *lpq* utility displays the contents of the print queues
- The *lpq* command format:

  `lpq` *options*

| Options | Function |
|---|---|
| + | Display the print queue until it empties |
| -P*printer* | Use *printer* instead of the default print queue |

Example:

```
% lpq +
```

# THE *lprm* UTILITY

- The *lprm* utility removes jobs from print queues
- Use the -P option to specify the printer

  `lprm [-P`*printer*`] `*job*

- *job* indicates the job to remove

  - Remove all jobs for the user

  *job#* Remove specific *job#*

  Examples:

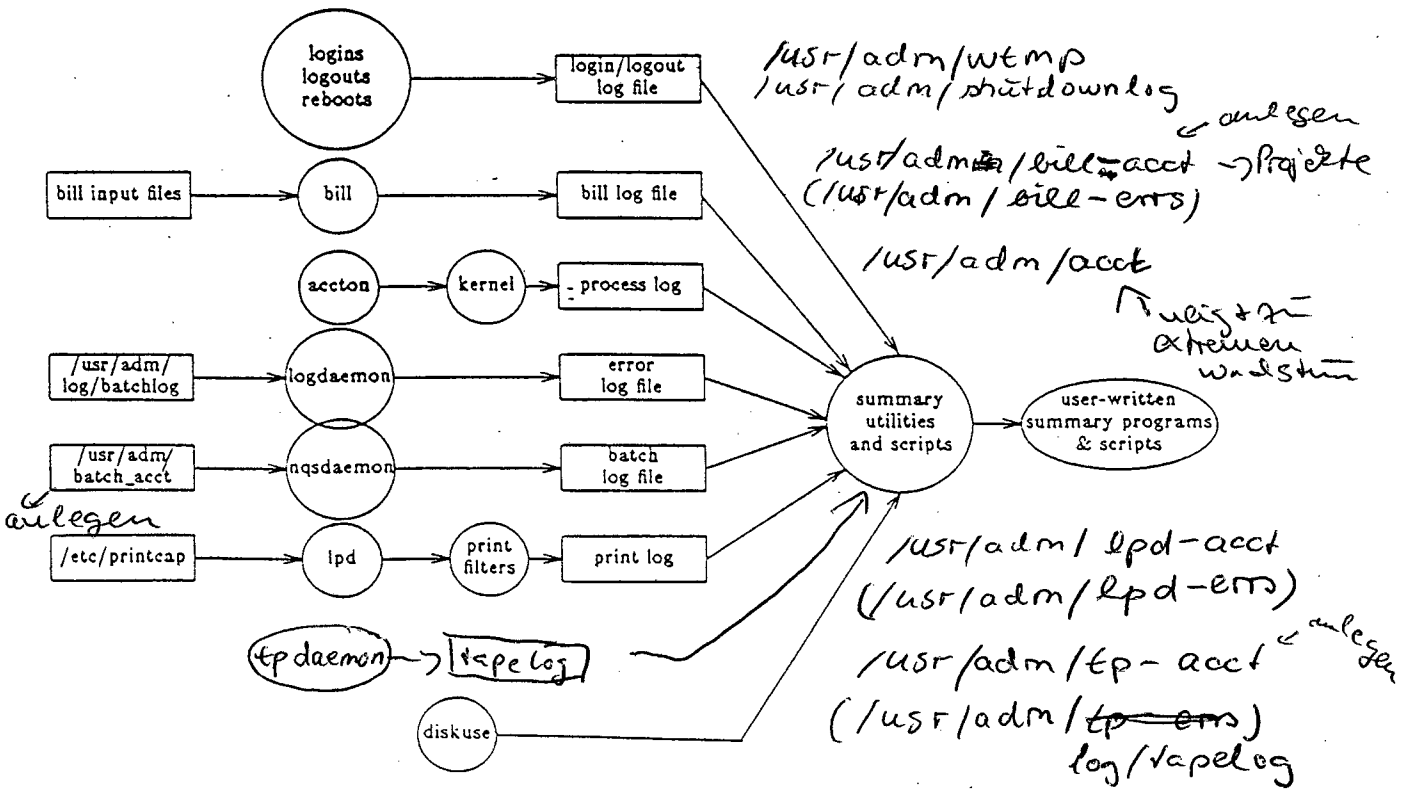
  ```
  % lprm -
  % lprm job#
  ```

# ACCOUNTING OVERVIEW

- Accounting keeps track of:
  - o Processes
  - o Logins
  - o Disk usage
  - o Tape usage
  - o Printer usage

- Each of the above has its own set of data files and programs to process the datafiles

- The programs are *sh* or *awk* scripts and can be modified to produce site-specific reports



# PROCESS ACCOUNTING

- Process accounting is initiated with */etc/accton* in the */etc/rc* file
- The command requires an argument:
  - o `/etc/accton /usr/adm/acct`
  - o *acct* is the binary data file which contains accounting information
- Data is catinated to */usr/adm/acct* until the disk space in the */usr/adm* file system exceeds 98% of capacity
- Accounting must be started manually when the available space drops back to 96%
- To turn off accounting manually, use:

  `/etc/accton`

- To turn off accounting permanently, comment the line in */etc/rc*.

# PROCESS ACCOUNTING (page 2)

- For each process, accounting records:
  - o Process name
  - o User, group, and activity IDs
  - o Total time run
  - o Total CPU time
  - o Total ~~user~~ System time
  - o ~~Total group time~~
  - o Start time
  - o TTY
  - o Memory usage in KByte * Sekunden
  - o Disk accesses Anzahl IO - count Block

# BUILDING REPORTS OVERVIEW

- The source to the *accounting* program is kept in the file */usr/adm/accounting.c*
- Executing */usr/adm/accounting* from */usr/lib/crontab* causes data to be generated for daily, weekly, and monthly reports

```
00 * * * * /usr/adm/accounting >
/dev/console
```

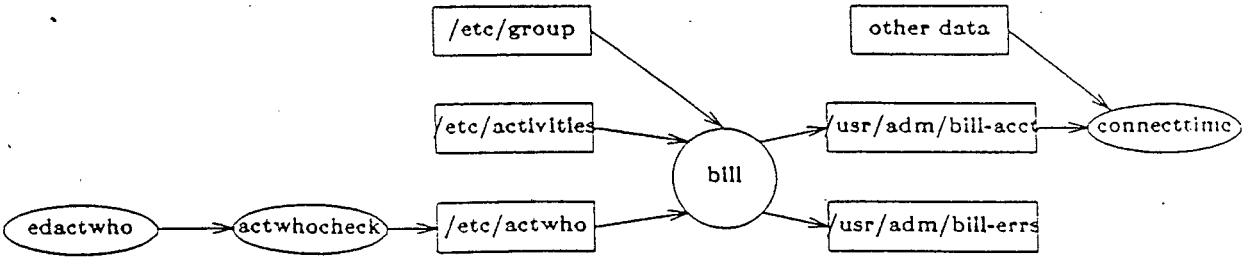
- When *accounting* finds that the day, week, or month has changed since the last time it ran accounting it invokes scripts to generate reports
- The *daily, weekly,* and *monthly* scripts may be edited to fit your site

# BILLING ACCOUNT OVERVIEW

- The accounting system is based upon users, groups, and activities
- Users belong to groups
- The tasks they do are called activities
- An account is a group paired with an activity
- To set up accounting by specific group, user, and activity, three files are required:

```
/etc/group
/etc/activities
/etc/actwho
```

- If these files are not set up, accounting uses the users primary group (from */etc/passwd*) and sets the activity to miscellaneous



# The */etc/group* File

- Each group must have a specific ID:

```
engineer:*:100:williams,george,smith
software:*:200:anderson,johnson,bryce
lp:*:300:george,johnson
diag:*:400:gsmith,james,knole,jessie
docs:*:500:larson
lang:*:600:larson,jensen
```

- A user can belong to 16 groups

# THE */etc/activities* FILE

- Billing activities are described in the file */etc/activities*
- */etc/activities* entries identify an activity's name and number
- The format of */etc/activities* entries is:

  *name*:*number*

  Example:

```
misc:0
backup:18
random:38
develop:58
support:78
lp:98
study:128
```

# THE */etc/actwho* FILE

- Each *actwho* entry lists which users and groups can bill to a particular activity
- Use the */etc/edactwho* command to edit this file
- The format of *actwho* entries is:
  *group.activity:user1,user2,...*
- An asterisk, which stands for all, can replace *group*, *activity*, or *user*
- A single-character wildcard, ?, can be used in *group* or *activity*

Example *actwho* file:

```
*.*:root
*.study:larson
*.backups:larson
engineer.develop:smith,george
engineer.support:williams,george
software.develop:anderson,johnson
software.support:bryce,johnson
lp.*:george,johnson
*.misc:*
```

# THE */usr/convex/bill* COMMAND

- The default group and activity are used by accounting unless the user uses the *bill* command to select a specific group and activity

- The billing group can be changed to any group ID belonging to the user (given in */etc/group* )

- The billing activity can be changed to activities listed in the */etc/activities* file

- The */usr/convex/bill* command changes the account billing

- The *bill* command format:

```
/usr/convex/bill group  activity
% bill software develop
```

- If no parameters are given, the current account billing is displayed

```
% bill
```

- If one parameter is given, it is assumed to be *activity*

```
% bill support
```

# ACCOUNTING SETUP

- After invoking */usr/adm/acct on* and */usr/adm/accounting*, zero-length files must be created and appropriate entries made in the "cap" files

- The line printer zero-length files are:

  */usr/adm/lpd-acct*
  */usr/adm/lpd-errs*

- Fields in */etc/printcap*:

  `af | if | pu`

- For tape allocation, create two zero-length files in */usr/adm*:

  *touch tp-acct tp-errs*

- To record changes of default *billing*, create the zero-length file *bill-acct* and to record *billing* errors, */usr/adm/bill-errs*

- To enable batch accounting (queue specific), use the *qmgr*:

  `set activity_id_offset=`*0-9*
  `set aid_mask=`*increment from /etc/activities*
  `set accounting=on`
  `set acc_logfile /usr/adm/batch-acct`

# PROCESS ACCOUNTING REPORTS

- *accounting* is run from */usr/lib/crontab* to determines which of the scripts to run

  */usr/adm/daily*

  */usr/adm/weekly*

  */usr/adm/monthly*

- The *daily*, *weekly*, and *monthly* shell scripts process the accounting records with the *sa* utility

- The scripts can be modified at a local site

# PROCESS ACCOUNTING REPORTS (page 2)

- *daily* copies data to daily{0-6}
  - o *sa* produces the appropriate *usracct{0-6}*, *savacct{0-6}*, *usracctw*, *savacctw*, *usracctm*, and *savacctm* files
  - o An ASCII daily report is created in the */usr/adm/daily.u* file

- *weekly* creates an ASCII report in the */usr/adm/week.u* file from *usracctw* and *savacctw*
  - o The appropriate weekly files are propagated - *u.w.mon.day.yr*, *s.w.mon.day.yr*

- *monthly* creates an ASCII report in the */usr/adm/month.u* file using the summary information in *usaracctm* and *savacctm*
  - o Monthly files are created - *u.m.month.yr.* and *s.m.month.yr*

# PROCESSING ACCOUNTING RECORDS

- *sa* and *lastcomm* will convert the binary */usr/adm/acct* records to ASCII

- Using the *s* option with *sa* saves information to 2 files:

| | |
|---|---|
| */usr/adm/savacct* | Contains accounting information for each process |
| */usr/adm/usracct* | Receives accounting information for each user and group/activity combination |

The *sa* command format:

```
/etc/sa options /usr/adm/acct
```

- Without options the format includes:

| | |
|---|---|
| number of executions | real time |
| cpu time | average I/O operations |
| core usage | command name |

# THE *sa* UTILITY (Continued)

- Some valid *options* are:

| | |
|---|---|
| c | Include percentage of total time |
| d | Sort by average number of disk I/O operations |
| D | Print and sort by average number of disk I/O requests |
| e | Display records as they appear in the accounting file |
| g | Specify by group |
| s | Save output in *usracct* and *savacct* |

- Fields in *usracct* and *savacct*

| | |
|---|---|
| CPU | Sum of user and system time |
| re | Real time |
| k | CPU-time averaged core usage |
| avio | Average I/O operations |
| tio | Total I/O operations |
| k*sec | CPU storage integral (1K storage * seconds) |
| u | User CPU time |
| s | System CPU time |

- The *options* selected determine which fields are displayed

# *sa* OUTPUT

- `/etc/sa -e` output includes:

  command executed
  starting time
  total CPU time used
  tty identification
  user ID
  group ID
  activity ID
  elasped time
  memory use
  system CPU time

```
rm 596130580 1.49 tty04   0  68 0 4.97s  41k*sec 155io  1.37
ls 596130615 0.14 tty04   0  68 0 0.31s  13k*sec   0io  0.07
cp 596130688 0.03 tty03 272  24 0 0.24s   0k*sec   4io  0.03
```

# PROCESSING REPORTS WITH *IDTONAME*

- Deciphers accounting data
- Converts user, group and activity ID's to a name
- Converts timestamp to date
- Writes to standard output
- Has the format:

```
idtoname -[aegtux]

a       convert to activity name
e       echo field
g       convert to group name
t       convert timestamp to ASCII date
u       convert to user name
x       skip field; do not print
```

- Specific fields can be sent to standard out:

```
/etc/sa -e | idtoname -eteeugaxxxx
```

- Displays command, time started, CPU time, user name, group name and activity name

# ACCOUNTING REPORTS

- The *daily.p*, *weekly.p* and *month.p* files are in ACSII format

- The general format of these report files:
  - o Number of command executions
  - o Real time
  - o CPU time
  - o Average I/O operations per execution
  - o Core usage
  - o Command name

- Files used to generate reports:

| | |
|---|---|
| daily | (*daily*{*0-6*}) |
| weekly | (*u.w.*{*mon.day.yr*}) |
| monthly | (*u.m.*{*mon.yr*}) |

- Copy the data file to another directory with the name *savacctw*:

```
cd /tmp
cp /usr/adm/daily5 savacctw
```

- Process the data with the command:

```
/etc/sa -Fw /dev/null > report
```

- Because *sa* requires an accounting file name */dev/null* is used here

# REPORT GENERATING *awk* SCRIPTS

- Accounting summary awk scripts reside in */usr/adm/sumscripts* man sumscripts
- Use these scripts to output reports
- Output from summary utilities can be piped through these scripts
- Pipe the following utilities through *awk* scripts:

```
(/etc/diskuse | awk -f
diskbygrp.awk)
```

| Utility | Summary Script |
|---|---|
| /etc/connecttime | connecttime.awk |
| /etc/diskuse | diskbygrp.awk |
| /etc/diskuse | diskusrgrp.awk |
| /etc/diskuse | diskmerge.awk |
| /etc/sa -g | sabyact.awk |
| /etc/sa -g | sabygrp.awk |

# REPORT GENERATING *awk* SCRIPTS (cont.)

- Scripts can process information in general log format:

```
(awk -f genbyact.awk /usr/adm/lpd-
acct)
```

| Utility | General Format Script |
|---|---|
| /usr/adm/lpd-acct | genbyact.awk |
| /usr/adm/lpd-acct | genbygrp.awk |
| /usr/adm/lpd-acct | genbygrpact.awk |
| /usr/adm/tp-acct | tape.awk |

- Four utilities can be executed without script conversion:

/etc/sa
/etc/diskuse
/etc/connecttime
/etc/pac

# *awk* SCRIPTS FOR *sa* OUTPUT

- Awk scripts take the raw ASCII data produced by *sa* and summarize it
- Two of the available scripts are:

  */usr/adm/sumscripts/sabyact.awk*
  */usr/adm/sumscripts/sabygrp.awk*

- *sabyact.awk* summarizes the information by account
- *sabygrp.awk* summarizes the information by group
- These scripts process the output of the *sa -g* command
- Example:

```
# /etc/sa -g | awk -f /usr/adm/ \
  sumscripts/sabygrp.awk
```

# *awk* SCRIPTS (page 2)

- The format of *sabygrp.awk* and *sabyact.awk* is:

```
group/acct commands CPU I/O_operations memory
```

Example *sabygrp.awk* output:

```
bin      9961   47.75cpu   22528tio   13833k*sec
staff    3221   15.42cpu   10031tio   10002k*sec
bin       961    9.91cpu    2419tio    5719k*sec
```

# CONNECT TIME ACCOUNTING

- The binary file */usr/adm/wtmp* contains records of logins and logouts
- The utility */etc/connecttime* displays the contents of *wtmp* in ASCII
- The format of *connecttime* output is:

```
event  time  tty  UID  GID  activity_id  remote
```

- Sample *connecttime* output:

```
started      542291939   ttyp0   754   49   0
terminated   542291540   ttyp0
started      542291497   ttyp0   754   49   0
started      542273627   tty37   124   67   0
started      542272987   tty36   124   67   0
terminated   542268519   ttyp0
terminated   542268276   tty37
started      542267058   ttyp0   59    53   0
terminated   542265726   tty36
```

  - o *event* is either started or terminated
  - o *remote* has no column in this example

# CONNECT TIME ACCOUNTING (page 2)

- The output of *connecttime* can be summarized by the awk script */usr/adm/sumscripts/connecttime.awk*

- To run *connecttime.awk*, enter:

```
# /etc/connecttime|\
 awk -f /usr/adm/sumscripts/connecttime.awk
```

- The *connecttime.awk* output format is:

```
connecttime length UID GID activity_ID TTY remote
```

Example *connecttime.awk* output:

```
542291497     43     754    49     0     ttyp0
542272987   17483    124    67     0     tty34
542267058    1461     59    53     0     ttyp0
542266242    1487    124    67     0     tty37
542263884    1704     -1    -1    -1     ttyp0
542263563      17      0    10     0     ttyp0
```

# DISK USE ACCOUNTING

- Disk use accounting is run through a combination of standard utilities, shell scripts and *awk* scripts
- The main disk use utility is */etc/diskuse*
- *diskuse* displays disk usage for a directory by user and group
- The *diskuse* command format:

  `/etc/diskuse -d` *directory*

# SUMMARIZING PRINTER USE

- The */etc/pac* script displays a summary of printer use
- Example:

```
# /etc/pac
```

- Example *pac* output:

```
USER        PAGES     RUNS      $COST
smith         6         2         0.12
jones        20         6         0.40
anderson      2         1         0.04
TOTAL        28         9         0.56
```

- Look at: man pac

# SUMMARIZING TAPE USE

- The *awk* script */usr/adm/sumscripts/tape.awk* summarizes the tape accounting information kept in */usr/adm/tp-acct*

- To run *tape.awk*, enter:

```
# awk -f /usr/adm/sumscripts/tape.awk \
    /usr/adm/tp-acct
```

- The *tape.awk* output format is:

```
time_allocated length UID GID activity_id tape_name
```

- Example output:

```
516668143    1         0      10    0    bogus
516668158    92        0      10    0    bogus
516668266    237408    0      10    0    bogus
516906228    1105      293    60    0    unit0
517678158    8710      75     53    0    unit0
517686901    156       0      58    0    unit0
517687061    872       0      58    0    unit0
517687937    146       0      58    0    unit0
```

# SYSTEM GENERATION

- System generation involves modifying and booting the operating system kernel

- Reasons for System generation:
  - o Creating non-standard swap partitions (7.0 and previous operating systems)
  - o Installing user-written device drivers
  - o Installing CONVEX UNIX kernel patch code
  - o Installing layered products with special device drivers (COVUEnet)

- System generation steps:
  - o Create a file in the */sys/sysgen* directory that matches your configuration
  - o Create the files for generation with the */sys/sysgen/sysgen* utility
  - o Use the *make* utility to create the operating system image
  - o Copy the newly-created image to the SPU disk and */vmunix*
  - o Reboot the system with the new kernel

# THE SYSTEM CONFIGURATION FILE

- The *sysgen* utility uses a system configuration file to determine how to generate the new kernel
- A template configuration file (*REL_C1, REL_C2*) exists in the */sys/sysgen/* directory
  - o Copy this file and customize it
- Sample files are *C1DEF* and *REL_C1.PROF* Do not use these as a template
- The configuration file contains:
  - o Configuration and option parameters
  - o Hardware specifications; only modified when adding user-written device drivers

# GLOBAL CONFIGURATION PARAMETERS

- The first several lines of the configuration file define parameters global to all system files

- Several of these parameters can be modified to customize a particular site
  - o Each parameter is defined on a separate line
  - o The parameters are:

| | |
|---|---|
| `machine c1` | The machine is either a C1 or C2 |
| `cpu "C-1"` | The CPU type is "C-1" or "C-2" |
| `ident` *conf_filename* | *config_filename* is the name of the global configuration file for the sysgen |
| `maxmemsize 512` | Megabytes of memory supported by vmunix image |
| `options` *features* | System options |

- Various operating system values are modified in a boot file

# PARAMETERS (Continued)

- Pseudo-devices should not be changed
- Source is set to either *yes* or *no*
- The last global configuration parameter is *config*
  - o *config* Defines the location of standard devices needed by the kernel
  - o The format of the *config* parameter is:

    `config vmunix root on da0a swap on` *device*
- The swap device is by default *da0b*
  - o Additional devices are added by the *swapon* command
  - o In version 7.0 operating system, a single swap device can be added in the SPU file */mnt/os/bootcmd.local*
  - o This single swap device replaces the default device */dev/da0b* (no *sysgen* required)
  - o In later versions of the operating system, multiple non-b partitions can be added as swap in the SPU file */mnt/os/bootcmd.local*

# HARDWARE SPECIFICATION SECTION

- The hardware specification section in the configuration file determines which device drivers are included in the operating system kernel
- *vmunix* is built with all supported devices
- If adding hardware (special device drivers) that do not appear in the global configuration, edit the hardware portion of the configuration file

# EXAMPLE CONFIGURATION FILE

```
machine         c1
cpu             "C-1"
ident           rel_32
mememsize       512

options

pseudo-device   nfs 1
pseudo-device   inet 1
pseudo-device   loop 1
pseudo-device   ether 1
source          yes

config          vmunix root on da0 swap on da0 and da1
                and da2 and da3 and da4 and . . dd0 and dd1
                and dd2 and dd3 and dd4 and dd5 and dd6 and
```

# CONFIGURATION FILE (page 2)

```
hardware
 ccu 7 type IOP
  multibus 0
    controller type DKC-001 at csr 0x3f0 int 2
    unit 0 type DKD-001
    unit 1 type DKD-001
    unit 2 type DKD-001
    unit 3 type DKD-001
    controller type DKC-001 at csr 0x3f0 int 2
    unit 0 type DKD-001
    unit 1 type DKD-001
    unit 2 type DKD-001
    unit 3 type DKD-001
    controller type PRC-001 at csr 0x000 int 1
    unit 0 type PRT-001
    controller type LAN-001 at csr 0x000 int 2
    unit 0 type ex
    controller type LAN-003 at csr 0x000 int 0
    unit 0 type NET-001
    controller type GPI-001 at csr 0x000 int 1
    unit 0 type GPD-001
    controller type VER-001 at csr 0x000 int 0
    unit 0 type PLT-001
    controller type MTC-001 at csr 0x000 int 0
    unit 0 type MTD-001
    unit 1 type MTD-001
    controller type LAN-002 at csr 0x000 int 0
     unit 0 type HYP-001
```

# CONFIGURATION FILE (page 3)

```
  controller type ACM-001 at csr 0x000 int 0
  unit 0 type tty
  unit 1 type tty
  unit 2 type tty
  unit 3 type tty
  unit 4 type tty
  unit 5 type tty
  unit 6 type tty
  unit 7 type tty
  unit 8 type tty
  unit 9 type tty
  unit 10 type tty
  unit 11 type tty
  unit 12 type tty
  unit 13 type tty
  unit 14 type tty
  unit 15 type tty
  unit 0 type tty

ccu 6 type HSP
  driver HEC-001 csr 0x3333
  channel 0 type HED-001

ccu 5 type VIOP
  vme 0
  controller type UDD-002 at csr 0x000 int 0
  unit 0 type tty
  controller type LAN-007 at csr 0x000 int 1
  unit 0 type ex
  controller type DKC-203 at csr 0x000 int 1
  unit 0 type DKD-208
  unit 1 type DKD-208
```

# THE */sys/sysgen/sysgen* UTILITY

- */sys/sysgen/sysgen* utility creates files for system generation
  - o Necessary directories and files are created
  - o The *sysgen* command format:
  - o `/sys/sysgen/sysgen` *config_filename*
- *sysgen* creates the directory */sys/config_filename* and the file */sys/config_filename/makefile*
- The makefile is used to create the operating system image files:
  - o Makefiles contain program and file dependencies for the *vmunix* and CCU's images
  - o Header files (.h) contain the devices that are compiled into the system
  - o The header file (_conf.h) contains the entry points for the driver

# CREATING THE IMAGE FILES

- The *make* utility creates the operating system images
- The first *make* compiles source files that are not up to date:
  - o # cd /sys/*config_filename*
  - o # make depend
- Create the operating system image file *vmunix* and *iop* :
  - o # make >& make.out
  - o Errors will be placed in the file *make.out*
- The newly created image file is moved to the */sys/config_filename/os* directory:
  - o # make install

# COPYING THE FILES TO THE SPU

- The newly created file is copied to the SPU to boot the new operating system
- Save a copy of the old kernel:
  - o Invoke a SPU shell by entering ^P from the console
  - o The keyswitch must be set to *LOCAL MAINTENANCE*
- From the SPU, enter the following commands:
  - o (spu)> cd /mnt/os
  - o (spu)> mv vmunix vmunix.save
  - o Terminate the SPU shell (^D)
- Copy the operating system image to the SPU by entering:
  - o # spu -w /mnt/os/vmunix < /sys/*config_filename*/os/vmunix

# BOOTING THE NEW KERNEL

- In order to use the new kernel reboot the new system
- Use *shutdown* to bring the system down to the SPU
- Reboot CONVEX UNIX
- If problems occur with the new kernel
  - o return to the SPU
  - o Restore the old OS image to */mnt/os/vmunix*
- When the new operating system kernel is working correctly copy the image file to the CONVEX UNIX file */vmunix* :
  - o # spu -r /mnt/os/vmunix > /tmp/hold
  - o # mv /tmp/hold /vmunix

# SECURITY FEATURES

- Each tty line can be set up to require an additional password
- A user named *dialin* must be added to the */etc/passwd* file
  - o A *dialin password* is required :
  - o Passwd entry:
    dialin::280:58::/:/bin/false
- The *passwd* command is used to assign the dialin passwd
- Each line is represented by a file in the */dev* directory
- The lines that are to require a dialin password start with *ttyd* rather than *tty[0-9]*

  # mv /dev/tty01 /dev/ttyd1 to rename device
- The line's */etc/ttys* entry must reflect the new name:

```
ttyd1   "/etc/getty D1200"   vt100n   on   secure dialup
ttyd2   "/etc/getty D1200"   vt100n   on   secure dialup
```

# SECURITY FEATURES (Continued)

- The system will automatically log failed login attempts if the file */usr/adm/badlogins* exists
- The following information is kept in *badlogins* :

  *time tty username*

- Example *badlogin* entries:

```
Fri Aug  7 07:39:30 1987 tty09 marker
Fri Aug  7 08:10:29 1987 tty0f drops
Fri Aug  7 08:33:18 1987 ttyd1 broggs
Fri Aug  7 09:09:24 1987 tty1e ff
Fri Aug  7 13:07:29 1987 tty3a smith
Fri Aug  7 13:41:58 1987 ttypb root
Fri Aug  7 14:16:28 1987 ttyp8 root
Fri Aug  7 14:20:23 1987 ttypb root
```

# ERASING DELETED FILES

- The operating system has the ability to erase deleted files
- If erasing is enabled, any blocks that are freed will be overwritten with a pattern
- The erasing feature is enabled by setting the operating system kernel variable `erase_unlink`
- The pattern to be written is kept in the operating system kernel variable `erase_pattern`
- Example entries for setting the above in */mnt/os/bootcmd.local* on the SPU

  tune cpu erase_unlink = 1

  tune cpu erase_pattern = FFFFFFFF

# LOGGING FAILED FILE ACCESSES

- The system has the facility to record information on each failed file access
- ASCII reports can be generated from the logfile
- Logging will discontinue if the filesystem becomes 98% full
- Logging is enabled and disabled with the */etc/faillogon command*
- *The faillogon* command format is:

  `/etc/faillogon` *filename*
- *filename* is the name of the file containing records of failed access attempts
  - o The file must exist for logging to work
  - o The file is */usr/adm/failure.log*
  - o If *filename* is not specified, failure logging is turned off

# THE */usr/adm/faillogpr* COMMAND

- The */usr/adm/faillogpr* command displays a report of failed file accesses
- The *faillogpr* command format is:

  `/usr/adm/faillogpr` *filename*

- *faillogpr* displays the following information on each failed file access:

  *time uid errno mode command file*

- *uid* will contain both the effective and real user IDs if they differ
- *errno* is the access error type as described in intro(2) in the Programmer's Reference Manual

# THE */usr/adm/faillogpr* COMMAND (page 2)

- *mode* is the access attempted
- *command* is the command used in the failed attempt
- *filename* is accurate at the time the report is created, not when the record was created

Example:

```
# /usr/adm/faillogpr /usr/adm/failure_log
Fri Aug 10 07:55:37 1990 skiles EACCES W csh /etc/passwd
Fri Aug 10 07:55:51 1990 anderson EACCES R grep /etc/rc
Fri Aug 10 07:56:17 1990 williams EACCES WX rm /mnt/smith/.
Fri Aug 10 07:56:40 1990 jones EACCES X csh /usr/adm/faillogpr
Fri Aug 11 19:08:54 1990 hopley EACCES X csh /mnt/hopley/calendar
#
```

# SECURITY AIDS

- Look for devices outside */dev* directory
- Check r/w permissions for disk device files, all setuid root files, */etc/passwd*, */etc/group*
- Log all super-user accesses
- Set up complete plan for which directories/files are accessible
- Compare *passwd* file from yesterday's every day
- Encourage users to use *crypt*
- Umask default to 077
- Run /etc/ncheck -s on the base system when it is installed
- Compare with later runs to see if any methods to break into your security have been used

# Index

## U



## V



## W