Contents

- Processes, priorities and signals Concepts.
- Redirection & Piping
- Apt-get utility
- Search
- Archiving

Processes

- Every program you run creates a process. For example
  - Shell
  - Command
  - An application

Processes

- System starts processes called **daemons** which are processes that run in the background and provide services

Every processes has a **PID**

- Every processes has a PID, positive integer between 2 and 32,768, The number 1 is typically reserved for the special init process
- When a process creates another, the first is the **parent** of the new process. The new process is called the **child process**. Parent waits for her child to finish

Process Priority

- Only process at a time may be executed on the CPU.
- Every process which is ready to run has a scheduling priority.
- The Linux process divides CPU time into time slices, in which each process will get a turn to run, higher priority processes first.
- User can affect the priority by setting the niceness value for a process

Process Priority Cont'd

- Niceness values range from -20 to +19, which indicates how much of a bonus or penalty to assign to the priority of the process.

- Most processes run with a niceness value of 0 (no change).

Process Priority Cont'd

- Smaller numbers are higher priority. Processes with a higher priority will run first in each time slice, and will run longer before its turn to run ends.

- Users can adjust this value down as far as +19 but can not increase it. Root can increase the priority of a process as high as -20

Adjusting Priority of a Process

- Adjusting process priority at invocation time

```
nice [-n adjustment] command
```

```
nice -n 10 makewhatis
```

- Adjusting the priority of a running process

```
renice priority [[-p] pid ...] [[-g] group ...] [[-u] user ...]
```

Signals

- A signal is a message sent to a process to perform a certain action
- Signals are identified by a signal number and a signal name, and has an associated action.
  - SIGTERM → 15
  - SIGKILL → 9

Viewing Processes

- Jobs show process in current terminal

Viewing Processes

- Process status command

`ps option(s)`

  - Output
    - PID
    - TTY -> terminal identifier
    - Execution time
    - Command name
  - Options
    - -e: all system processes
    - -f: full information
    - -u uid: display processes of that user.

Viewing Processes

Viewing processes with `top` utility

Searching for a Process

- Using `pgrep` command

```
pgrep option(s) pattern
```

```
$pgrep lp
```

- Options

-x: exact match

-u uid: processes for a specific user

-l: display the name with pid

Sending a Signal to a Process

- Using kill command
  - Default signal 15

```
kill [-signal] PIDs
```

```
Kill -l5  10345
```

- Examples

```
$pgrep -l mail
215 sendmail
12047 dtmail
```

Sending a Signal to a Process


```
$kill 12047


$pgrep -l mail
215 sendmail
```

- Using `pkill` command

```
pkill [-signal] process_name
```

- Example

```
$pkill -9 dtmail
```

Examples


```
sleep 500 &
[1] 3028
[1] + Done
jobs
[1] + Running     sleep 500&
fg 1
sleep 500
sleep 500
Ctrl+Z [1] + Stopped (SIGTSTP) sleep 500
jobs
[1] + Stopped (SIGTSTP) sleep 500
```

Examples Cont'd


```
bg %1
[1] sleep 500&
jobs
[1] + Running sleep 500&
kill -STOP %1
jobs
[1] + Stopped (SIGSTOP) sleep 500&
kill %1
[1] + Terminated sleep 500&
jobs
```

Standard Input and Output

- **Standard input**
  - Refers to the data source from which data is input to a command
  - Typically the keyboard
- **Standard output**
  - Refer to data destination to which data from the command is written
  - Typically the screen

Standard Input and Output

- **Standard error**
  - Refer to the output destination for the errors and messages generated by the command
  - Typically the screen also

Redirecting Input and Output

- `Command > fname`
- `Command >> fname`
- `Command < fname`

— Example

- `$ find /etc –name passwd > findresult`
- `$ ls –l /etc >> findresult`
- `Mail < file1`

Redirecting Standard Error

- Standard error is redirected to a file using the regular output redirection operator, but you must place a 2 in front of the operator (2>).
- Example
  - $ find / -name passwd 2> errs
  - $ find / -name passwd 2> errs > results

Using Pipe to connect Processes

- The pipe
  - A pipe (|) is used to send the output of one command as the input to another
  - The most common use of a pipe is to take a command that's output might go on for pages (such as cat or ls -l) and feed it through more.
  - Example
    - `$ ls –lR / | more`

APT-get (Advanced Packaging Tool)

- Ubuntu's package management system is derived from the same system used by the Debian GNU/Linux distribution.
- The package files contain all of the necessary files, meta-data, and instructions to implement a particular functionality or software application on your Ubuntu computer.
- The software management tools in Ubuntu will check dependencies automatically.

APT-Get

- The apt-get command is a powerful command-line tool used to work with Ubuntu's Advanced Packaging Tool (APT) performing such functions as
  - Installation of new software packages
  - Upgrade of existing software packages
  - Updating of the package list index
  - And even upgrading the entire Ubuntu system.

APT-Get Cont'd

- Install a Package

```
sudo apt-get install ksh
```

- Remove a Package

```
sudo apt-get remove ksh
sudo apt-get purge nmap
```

- * You may specify multiple packages to be installed or removed, separated by spaces.

APT-Get Cont'd

- Update the Package Index
  - Update is used to resynchronize the package index files from their sources.
  - The indexes of available packages are fetched from the specifies location(s) in /etc/apt/sources.list.
  - An update should always be performed before upgrade.

```
sudo apt-get update
```

APT-Get Cont'd

- Upgrade package
  - Install the newest versions of all packages currently installed on the system from the sources found in /etc/apt/sources.list.
  - An update must be performed first so that apt-get knows the new versions of packages available.

```
sudo apt-get upgrade
```

Finding Files with locate

- The locate command searches through a pre-built database containing the contents of your filesystem at the time the database was last updated.
- The locate database is built by using the `updatedb` command.
- Example
  - `locate passwd`

Locating Files with find

- The find command searches the live filesystem.
- find is slower than locate, causes more of a load on the system, but more powerful than locate.
- You are also limited by your own permissions.

Locating Files with find Cont'd

Expression	Definition
-name filename	Finds files matching the specified filename. Metacharacters are acceptable if placed inside " ".
-size [+|-]n	Finds files that are larger than +n, smaller than -n, or exactly n. The n represents 512-byte blocks.
-atime [+|-]n	Finds files that have been accessed more than +n days, less than -n days, or exactly n days.
-mtime [+|-]n	Finds files that have been modified more than +n days ago, less than -n days ago, or exactly n days ago.
-user loginID	Finds all files that are owned by the loginID name.
-type	Finds a file type, for example, f (file) or d (directory).
-perm	Finds files that have certain access permission bits

The sort command

- The sort command sorts text data after accepting it from either a file or the output of another command.

- Examples:
  - sort –t : -k1 /etc/passwd
  - sort –t : –k3 /etc/passwd
  - sort –t : -n –k3 –o passwd_sorted /etc/passwd

The tee command

- The `tee` command reads from the standard input and writes to the standard output and a file
- Example
  - `$ ls –lR / | tee fname | more`

Introduction To Archiving

- To safeguard your files and directories, you can create a copy, or archive, of the files and directories on a removable medium, such as a cartridge tape. You can use the archived copies to retrieve lost, deleted, or damaged files.

Archiving Files

- `tar` command archives files to and extracts files from a single file called a tar file.
- The default device for a tar file is a magnetic tape device.
- tar functions archivefile filenames
  - Function
    - c: create a new tar file
    - t: list table of content
    - x: extracts files from the tar command
    - f: specify the archive file
    - v: verbose mode

Archiving Files

- Example
- `tar cvf file.tar file1 file2 file3`

file1

file2

Archiving Files Cont'd

- Viewing an archive
  - tar tf file.tar
    - file1
    - file2
    - file3
- Extracting files from archive
  - tar xvf file.tar
    - file1
    - file2
    - file3

Compress Command

- Compression reduces a text file by 50 percent to 60 percent.
- compress [-v] filename
- Compress command replaces the original file with a new file that has a .Z extension.
- Example
  - compress -v files.tar
    - files.tar: Compression: 70.20% --
    - replaced with files.tar.Z

zcat Command

- zcat filename.Z

- Example
  - zcat file1.Z

uncompress Command

- uncompress options filename
- Example
  - uncompress -v files.tar.Z
    - files.tar.Z:-- replaced with files.tar

bzip2 Command

- The bzip2 command reduces the size of files.
- The original file is replaced by a file with the same name and a .bz2 extension.
- bzip2 [-v] filenames
- Examples:
  - bzip2 file1 file2 file3 file4
  - ls *.bz2
    - file1.bz2 file2.bz2 file3.bz2 file4.bz2

bzip2 Command Cont'd

- Restoring bzip2 file using the bunzip2 command
- Example
  - bunzip2 file1.bz2

bzcat Command

- bzcat command display the content of files compressed by bzip2

  —bzcat filename.bz2

Checking Free Space

- The `df` command displays number of free disk blocks and files.

```
df [-h]
```

- Example

```
df –h
Filesystem size used avail capacity Mounted on
/dev/hda0  15G  976M  14G   6%       /
```

Checking Free Space

- The du command display the total sum of space allocated to all files hierarchy rooted in the directory specified.

```
du [-sh] [dir…]
```

- Example

```
du –sh
   14K
```

System Shutdown

- It only requires reboot or shutdown when you need to
  - Add or remove hardware
  - Upgrade to a new version of Ubuntu
  - Or upgrade your kernel

  - `shutdown time`
  - `poweroff`
  - `Init 0`
  - `halt`

System reboot

- `shutdown -r`
- `reboot`
- `init 6`
- Press `CTRL+ALT+DEL`

Virtual Consoles

- Accessed with Ctrl-Alt-F_key
- Consoles 1-6 accept logins
- X server starts on the console F1
- Console from F2 to F6

Inode

- Linux see all files as numbers called “inodes”, or index nodes.

- Within each each filesystem is an inode table, in which all of the used inodes are mapped to particular files.

Inode Cont'd

- The information stored in this table for each entry includes the following:

1. The type of file
2. The file's permissions
3. The number of links
4. The file owner's user ID
5. The group owner's GID
6. When the file was last changed
7. When the file was last accessed
8. Where the file is on the media

Inode Cont'd

- To view inode number of a file
  - `ls -i fname`
    - 10978 fname

File Manipulation and Inodes

- The cp command
  - Allocates a new inode number for the copy, placing a new entry in the inode table
  - Creates a directory entry, referencing the file name to the inode number within that directory

File Manipulation and Inodes Cont'd

- Example
  - `ls -i f1`

1196100 f1

  - `cp f1 f2`
  - `ls -i f1 f2`

1196100 f1

1196463 f2

File Manipulation and Inodes Cont'd

- Inodes and the mv command
  - If the destination is on the same file system as the source:
  - mv creates a new directory entry with the new file name
- Example
  - `ls -i f1`

1196100 f1

  - `mv f1 f2`
  - `ls -i f2`

1196100 f1

Utilizing Links

- Soft Link (Symbolic Link)
  - The content of this entry is the path to the original file.
  - This allows you to use symbolic links across partition boundaries.
  - If you delete the original file, you end up with an “orphaned link”

Utilizing links to make shortcuts Cont'd

- Example

```
ls -l testfile
```

-rw-rw-r-- 1 user user 12 Mar 12 03:50 testfile

```
ln -s testfile testlink
```

```
ls -l testfile testlink
```

-rw-rw-r-- 1 user user 12 Mar 12 03:50 testfile

lrwxrwxrwx 1 user user 8 Mar 12 09:50 testlink → testfile

Utilizing links to make shortcuts Cont'd

- Hard Links
  - Instead of creating a new file, the new link (a new directory entry) is added to the appropriate directory file name listing, referencing the exact inode as the original file. Thus, the file only exists once, but in two places.
  - In the inode table, the link count is incremented.
  - Every filesystem has inodes that start counting from zero. A hard link cannot reach across partition boundaries. It can only exist within a single partition or media.

Utilizing links to make shortcuts Cont'd

- Example
  - `ls -l testfile`

-rw-rw-r-- 1 user user 12 Mar 12 03:50 testfile

  - `ln testfile testlink`
  - `ls -l testfile testlink`

-rw-rw-r-- 2 user user 12 Mar 12 03:50 testfile

-rw-rw-r-- 2 user user 12 Mar 12 09:50 testlink