UNIX Fundamentals
Part I

Presented By:

Gerry Hounsell

FACILITIES

- □ Training Hours
- □ Building Hours
- □ Restrooms
- □ Meals
- □ Telephones
- □ Messages
- □ Smoking Policy

INTRODUCTIONS

❑ Name

❑ Company

❑ Title/Function

❑ Job Responsibility

❑ UNIX Experience

❑ Expectations From Training

UNIX Fundamentals

COURSE AGENDA PART I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ Access to UNIX Systems
- □ Processes
- □ Filesystems & Directories
- □ Devices

UNIX Fundamentals
COURSE AGENDA PART II

- □ vi Editor
- □ Redirection & Regular Expressions
- □ Scheduling
- □ Logical Volume Management
- □ Networking
- □ Printing
- □ UNIX Tools & Utilities
- □ System Error Reporting
- □ Manual Pages

UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ Access to UNIX Systems
- □ Processes
- □ Filesystems & Directories
- □ Devices

UNIX History

- ❑ 1960’s - **Multi**plexed **I**nformation and **C**omputing **S**ervice (Multics).
- ❑ Ken Thompson continued to develop on the GE-645 Mainframe and wrote a game!
- ❑ Further development experience in DEC PDP-7 Assembly Language and work with Multics gave rise to **Un**iplexed **I**nformation and **C**omputing **S**ystem (Unics)
- ❑ Later renamed to UNIX

UNIX History

- ❑ 1970 – The Epoch
- ❑ Unix was officially named and ran on the PDP 11/20
- ❑ First funding for the development of UNIX from Bell Labs
- ❑ 1973 - UNIX rewritten in C to make the code portable to other machines
- ❑ Numerous versions of UNIX developed (1-7)

UNIX History

- ❑ 1980’s
- ❑ AT&T developed UNIX System III
- ❑ In 1982 AT&T combined UNIX 1-7 into UNIX System V Release 1
- ❑ These commercial versions no longer included source code, the University of California, Berkeley (UCB) continued to develop BSD Unix as an alternative
- ❑ BSD contribute TCP/IP network code into the mainstream UNIX kernel

UNIX History

- ❑ Further UNIX derivatives developed, mainly System V based
- ❑ 1982 SunOS developed
- ❑ 1980 Microsoft announced Xenix, the first UNIX for 16-bit microcomputers
- ❑ 1983 the Santa Cruz Operation (SCO) ported this to the Intel 8086 processor

UNIX History

❑ 1980’s and 1990’s – The UNIX Wars

❑ 1984 X/Open founded to standardise UNIX

❑ 1987-89 AT&T and Sun Microsystems merged Xenix, BSD, SunOS and System V into System V Release 4 (SVR4)

❑ 1990 Open Software Foundation released OSF/1 based closely on BSD UNIX

UNIX History

- 2000 - The dot-com crash
- Lead to significant consolidation of UNIX users
- Main derivatives of UNIX are now:
  - **IBM’s AIX**
  - **Sun Microsystems's Solaris**
  - **HP’s HP-UX**
  - **Linux (many different Vendors)**

UNIX History

“Some Key Features/Factors”

- Multi-tasking, Time Sharing
- Multi-user
- Network Capabilities
- Portability
- Flexibility
- Software Available
- Virtual Memory
- Case Sensitivity

UNIX's philosophy is the same as the C language's:

IT ASSUMES USERS KNOW WHAT THEY ARE DOING!!!!

UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ Access to UNIX Systems
- □ Processes
- □ Filesystems & Directories
- □ Devices

The Many Flavours of UNIX

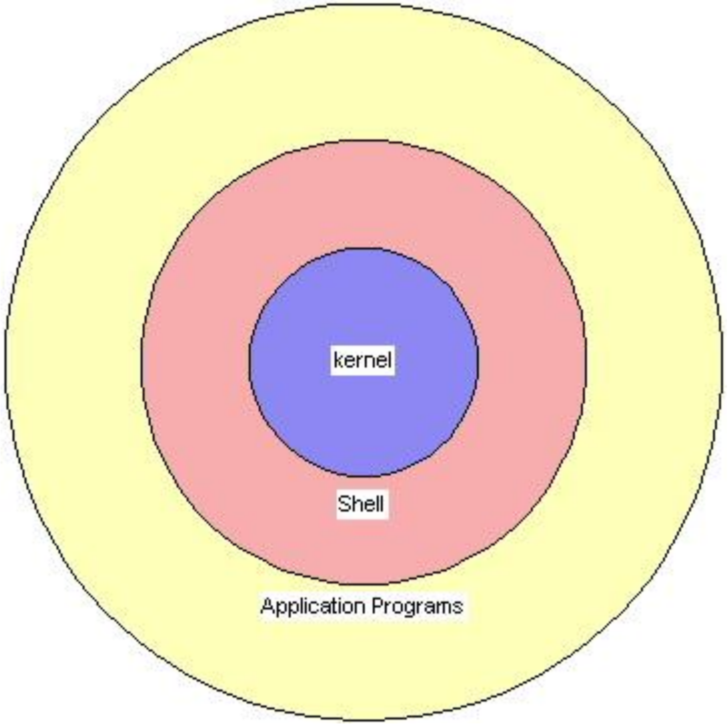
- Main derivatives of UNIX are now:
  - **IBM - AIX**
  - **Sun Microsystems - Solaris**
  - **Hewlett Packard - HP-UX**
- Linux
  - **Novell – SUSE**
  - **Red Hat**
  - **Mandriva**
  - **Debian**
  - **Ubuntu**

UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
  - **The Kernel**
  - **The Shell**
  - **Filesystems**
- □ Access to UNIX Systems
- □ Processes
- □ Filesystems & Directories
- □ Devices

The Structure of UNIX

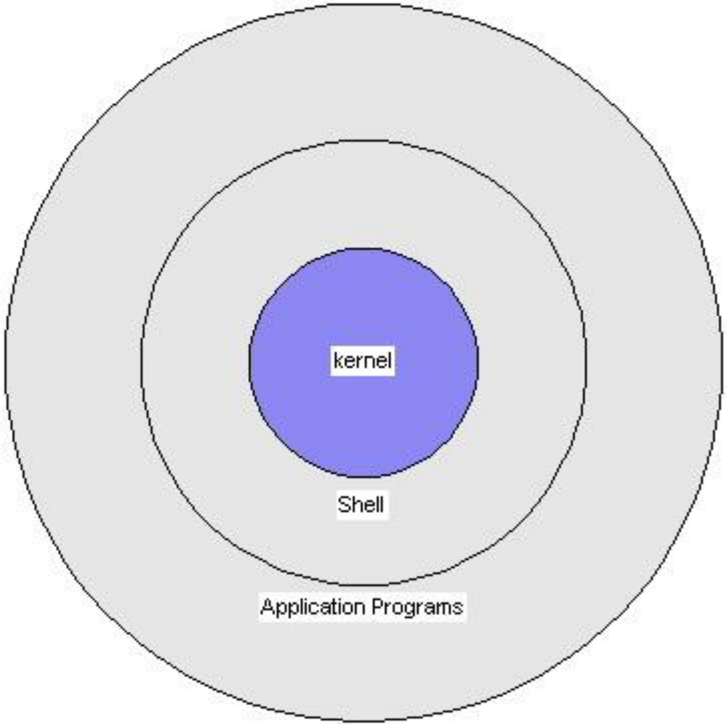
- ❑ The Kernel
- ❑ The Shell
- ❑ Filesystems



UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
  - **The Kernel**
  - **The Shell**
  - **Filesystems**
- □ Access to UNIX Systems
- □ Processes
- □ Filesystems & Directories
- □ Devices

The Kernel

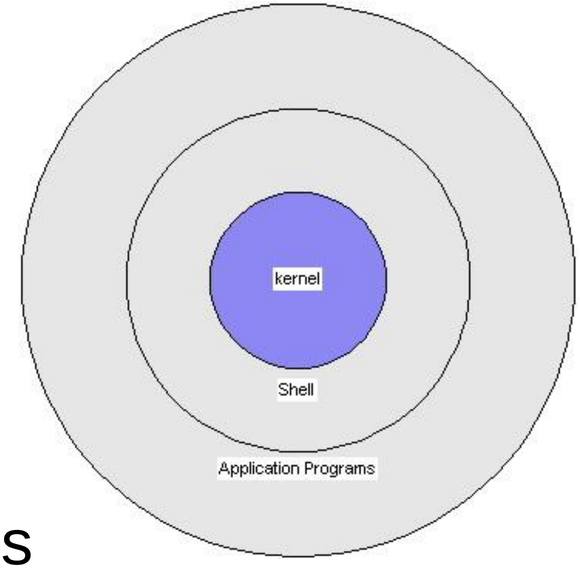


The kernel is the direct link to the hardware and controls all I/O requests.

The shell is a high-level link to the kernel.

The application layer normally sits above the shell.

The Kernel



- ❑ Manages the files on disk
- ❑ Schedules multiple processes / users
- ❑ Written in “c” with a ‘small’ amount’ of machine dependent assembler

UNIX Fundamentals Part I

□ UNIX History

□ The Many Flavours' of UNIX

□ The Structure of UNIX

- **The Kernel**
- **The Shell**
- **Filesystems**

□ Access to UNIX Systems

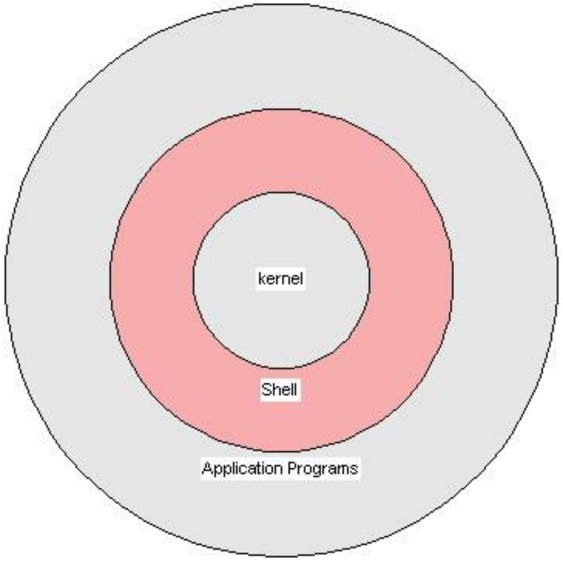
□ Processes

□ Filesystems & Directories

□ Devices

The Shell - I

- ❑ The UNIX shell is a command interpreter
- ❑ A shell is a program that interprets and runs the commands typed at the console by the user. The shell sends requests to the kernel, which executes them.
- ❑ The three ‘best known’ shells are
  - **C shell (csh)**
  - **Bourne shell (bsh)**
  - **Korn shell (ksh)**
- ❑ Each shell can be programmed.



The Shell -II

- The Shell is similar to command line in DOS
- The file separator in UNIX is a *forward slash (/)* and NOT a *back slash (\)* this has a special meaning/use.
- The Shell is case sensitive: myfile, MYFILE and MyFile are three different names.
- The Shell has very few built in commands to do the most basic tasks, e.g.
  - **cp – copy.**
  - **mv – move.**
  - **rm – delete.**
  - **echo – print to screen.**
  - **clear – clear the screen.**
  - **cal – display calendar.**
  - **Date – display or change the date.**

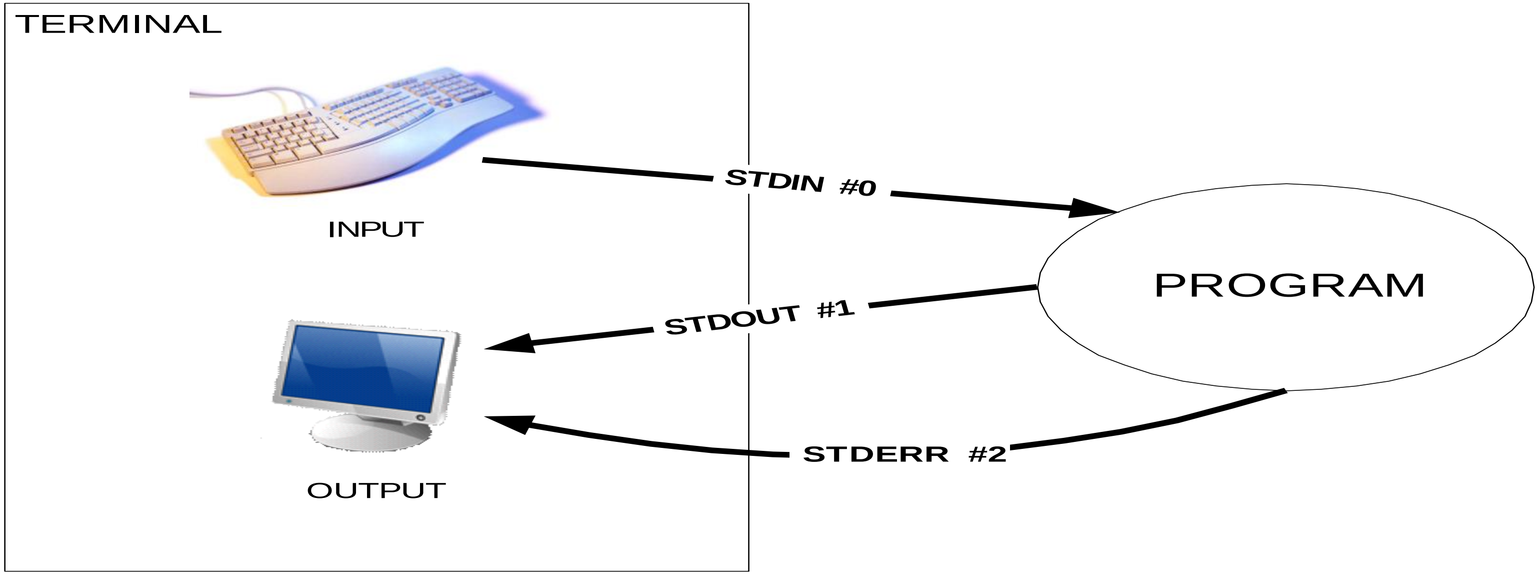
The Shell - III

- ❑ Programs containing shell commands are called shell scripts.

- ❑ Shell scripts are important because they:
  - ▪ **are an easy way of adding new commands.**
  - ▪ **allow prototyping.**
  - ▪ **allow on-off systems to be developed easily.**

- ❑ The above are only possible because the standard shells provide a complete programming language.

The Shell - IV



- ❑ Three files are automatically opened for each process in the system. These are:-
  - **standard input**
  - **standard output**
  - **standard error**
- ❑ These defaults can be changed using redirection.

The Shell V

- The default file descriptors can be changed using redirection.
- Redirection input from a file: **<**
  - **Command < filename**
- Default Standard output:
  - **Outputs to screen**
- Redirect output from a file: **>**
  - **command > filename**
- Redirecting & appending output to a file: **>>**
  - **command >> filename**

The Shell - VI

- Shell variables
  - **Define your environment**
    - ***HOME directrory***
    - ***TERMinal type***
    - ***Search PATH***
  - **Additional variables can be defined**
- Use the export command to declare the variable
  - **ORAHOME=/home/oracle/9.1/**
  - **export ORAHOME**
- Use **set** or **env** commands to display the shell variables in your current environment.
- Use **unset** to delete or remove the variable from the environment.

The Shell - VII

❑ Quoting Metacharacters

- **‘ ‘ Single Quotes: ignores the special meaning of all metachracters between the quotes**
- **“ “ Double Quotes: Ignores the special meaning of all metacharacters except for $, ` and **
- **\ Backslash: Ignores the special meaning of the following character.**

The Shell VIII

- A shell script is a collection of commands stored in a text file.
- Shell scripts can be invoked in three ways:
  - **$ ksh <scriptname>**
  - **$ <scriptname>**
  - **$. <scriptname>**
- A command returns a value to the parent process. By convention, zero means success and a non – zero means an error occurred.
- Commands in a pipeline return a single value to their parent.
- The environment variable $? Contains the return code of the previous command. e.g.
  - **0 = success**
  - **1 = worked but not successful**
  - **2 = failure**
  - **127 = program does not exist in directory**

UNIX Fundamentals Part I

- ❑ UNIX History
- ❑ The Many Flavours’ of UNIX
- ❑ The Structure of UNIX
  - **The Kernel**
  - **The Shell**
  - **Filesystems**
- ❑ Access to UNIX Systems
- ❑ Processes
- ❑ Filesystems & Directories
- ❑ Devices

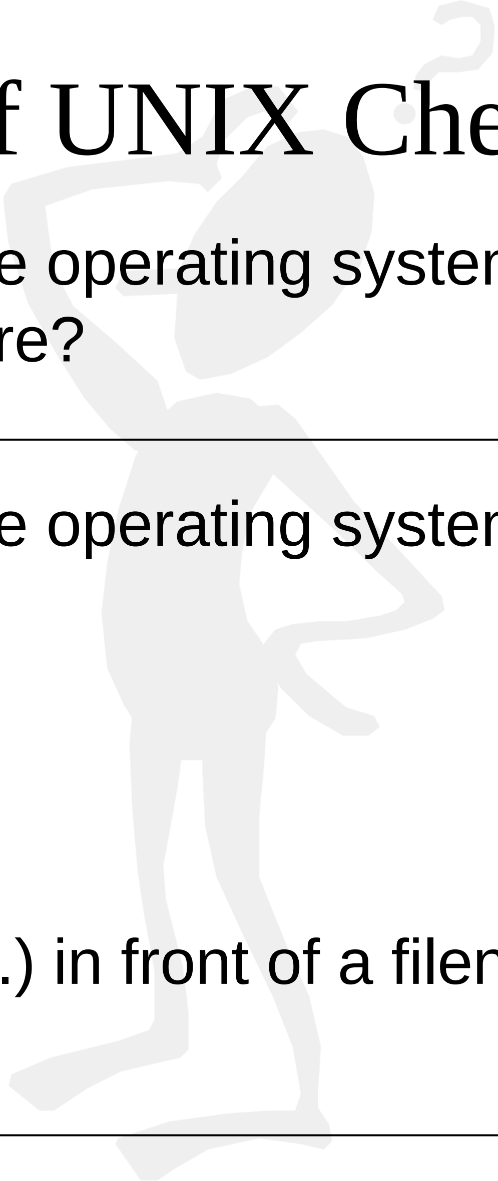
Filesystems

- ❑ The UNIX File System manages and controls access to files and directories.
- ❑ UNIX keeps track of opened and closed files, and manages files on the hard disk.
- ❑ The filing structure of the UNIX operating system is hierarchical.
- ❑ It is a tree structured system, completely open (assuming necessary permissions) to every user on the system, with everything emerging from / (root) at the top.

Filesystems

- ❑ Naming Conventions
  - **Simple, any ASCII characters can be used**
  - **It is recommended that no ‘metacharacters’ are used**
  - **It is recommended to use only**
    - ***Letters***
    - ***Digits***
    - ***Underscore (_)***
    - ***Hyphen (-)***
    - ***Dot (.)***
- ❑ Filenames/Directories beginning with a dot (**.**) are hidden.

Structure of UNIX Checkpoint - 1



- Which part of the operating system interacts directly with the hardware?

- Which part of the operating system does the user interact with?

  a) Shell

  b) Kernel

  c) Filesystem

- What does the (.) in front of a filename or directory name mean?

Structure of UNIX Checkpoint - 2

- Name the three file descriptors assigned by the shell when a program starts?

  1. ....

  2. ....

  3. ....

- What characters are not recommended to be used when naming files & directories?

UNIX Fundamentals Part I

❑ UNIX History

❑ The Many Flavours' of UNIX

❑ The Structure of UNIX

❑ Access to UNIX Systems

- **Logging on and off**
- **whoami & whereami**
- **Changing Users**

❑ Processes

❑ Filesystems & Directories

❑ Devices

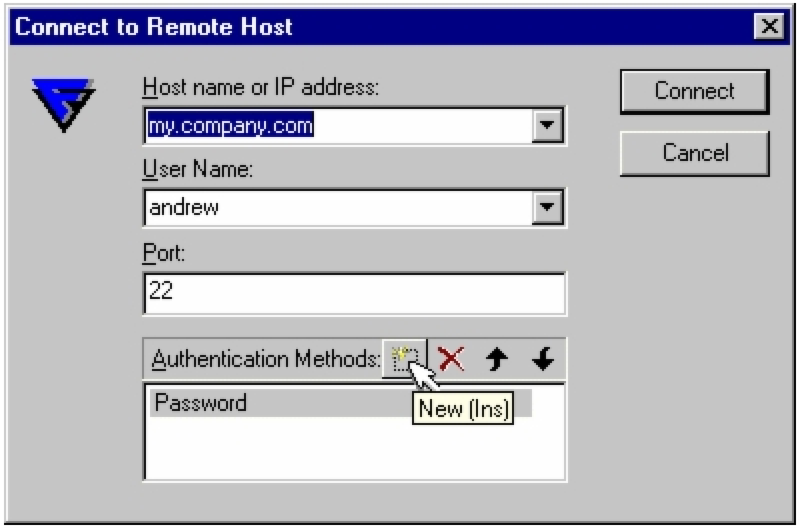
Accessing UNIX Systems

- Logon requires a username and password
- UNIX is cAsE sensitive for both username and password
- .profile used to initialise user preferences during logon
- You can use telnet or ssh to connect to a UNIX system.

Logging on & off

- ❑ You will need a username & password
- ❑ You will need an ssh client
  - **Putty**
  - **F-Secure Client**
- ❑ A UNIX host to connect to!
  - **Example:**

    naac01.systems.uk.hsbc



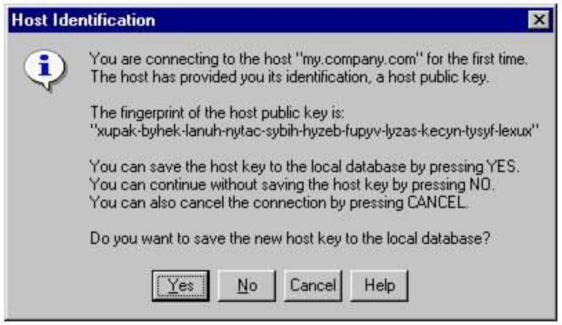
Connect to Remote Host

Host name or IP address: my.company.com

User Name: andrew

Port: 22

Authentication Methods: Password



Host Identification

You are connecting to the host "my.company.com" for the first time.
The host has provided you its identification, a host public key.

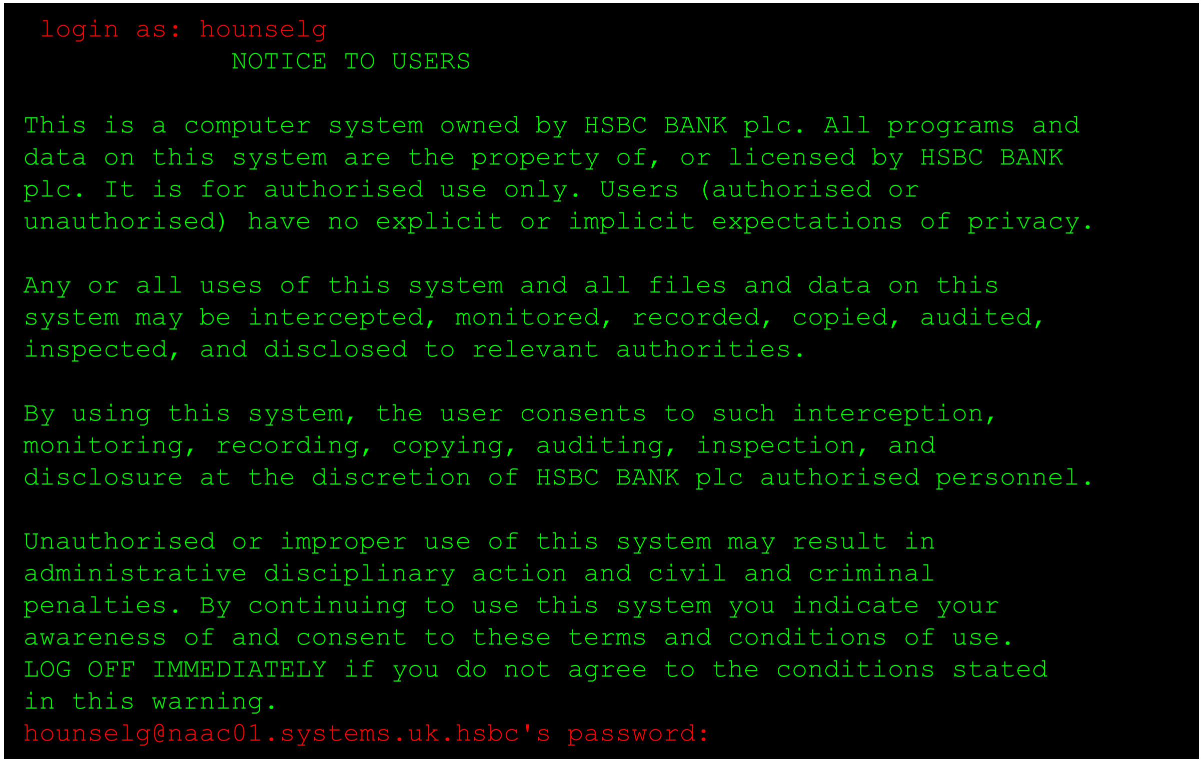
The fingerprint of the host public key is:
"xupak-byhek-lanuh-nytac-sybih-hyzeb-fupyv-lyzas-kecyn-tysyf-lexux"

You can save the host key to the local database by pressing YES.
You can continue without saving the host key by pressing NO.
You can also cancel the connection by pressing CANCEL.

Do you want to save the new host key to the local database?

Logging on & off

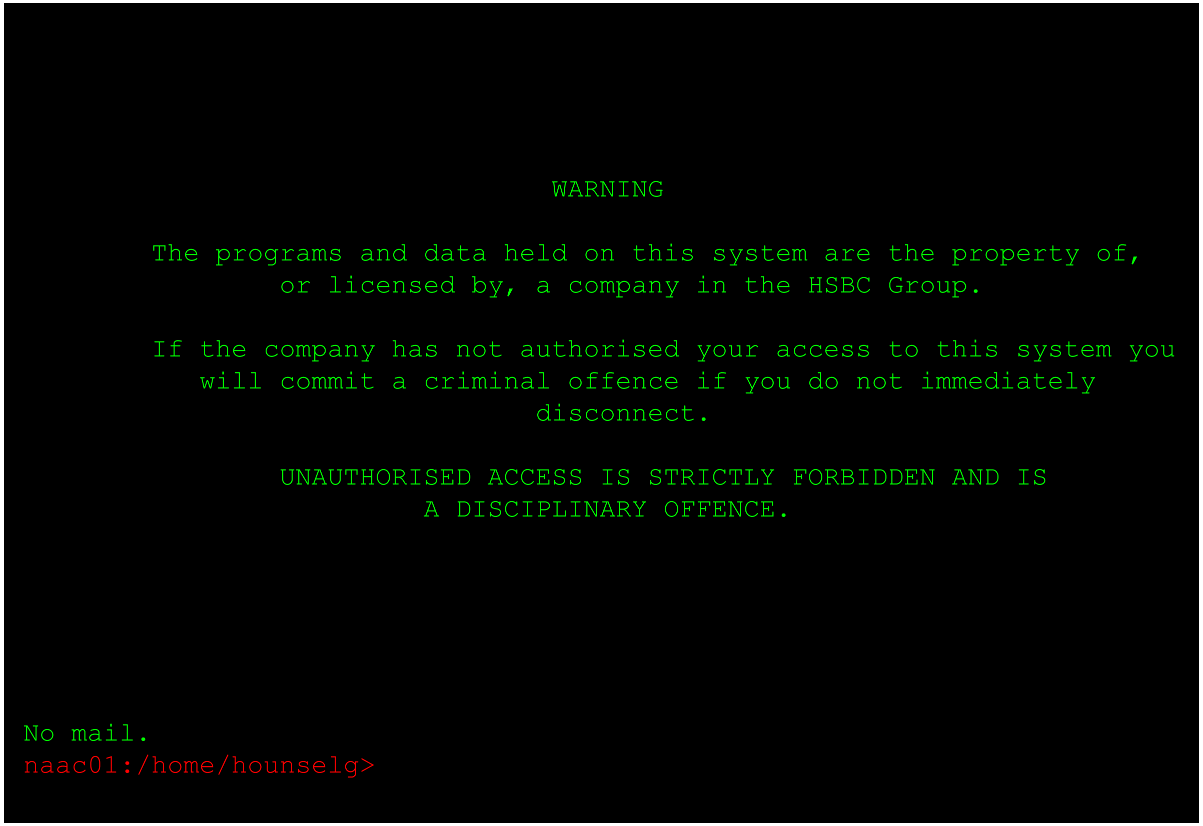
- Logging on:



```
 login as: hounselg
             NOTICE TO USERS

This is a computer system owned by HSBC BANK plc. All programs and
data on this system are the property of, or licensed by HSBC BANK
plc. It is for authorised use only. Users (authorised or
unauthorised) have no explicit or implicit expectations of privacy.

Any or all uses of this system and all files and data on this
system may be intercepted, monitored, recorded, copied, audited,
inspected, and disclosed to relevant authorities.

By using this system, the user consents to such interception,
monitoring, recording, copying, auditing, inspection, and
disclosure at the discretion of HSBC BANK plc authorised personnel.

Unauthorised or improper use of this system may result in
administrative disciplinary action and civil and criminal
penalties. By continuing to use this system you indicate your
awareness of and consent to these terms and conditions of use.
LOG OFF IMMEDIATELY if you do not agree to the conditions stated
in this warning.
hounselg@naac01.systems.uk.hsbc's password:
```

Logging on & off

- Logged on:



```
                              WARNING

         The programs and data held on this system are the property of,
                 or licensed by, a company in the HSBC Group.

         If the company has not authorised your access to this system you
            will commit a criminal offence if you do not immediately
                                 disconnect.

                 UNAUTHORISED ACCESS IS STRICTLY FORBIDDEN AND IS
                          A DISCIPLINARY OFFENCE.


No mail.
naac01:/home/hounselg>
```

Logging on & off

- Logging off:
  - **Three easy ways to *SAFELY* log off**
    1. ***Type the command exit at the command prompt:***

```
naac01:/home/hounselg> exit
```

2. ***Use the key combination ctrl - d***
3. ***Type the command logout at the command prompt:***

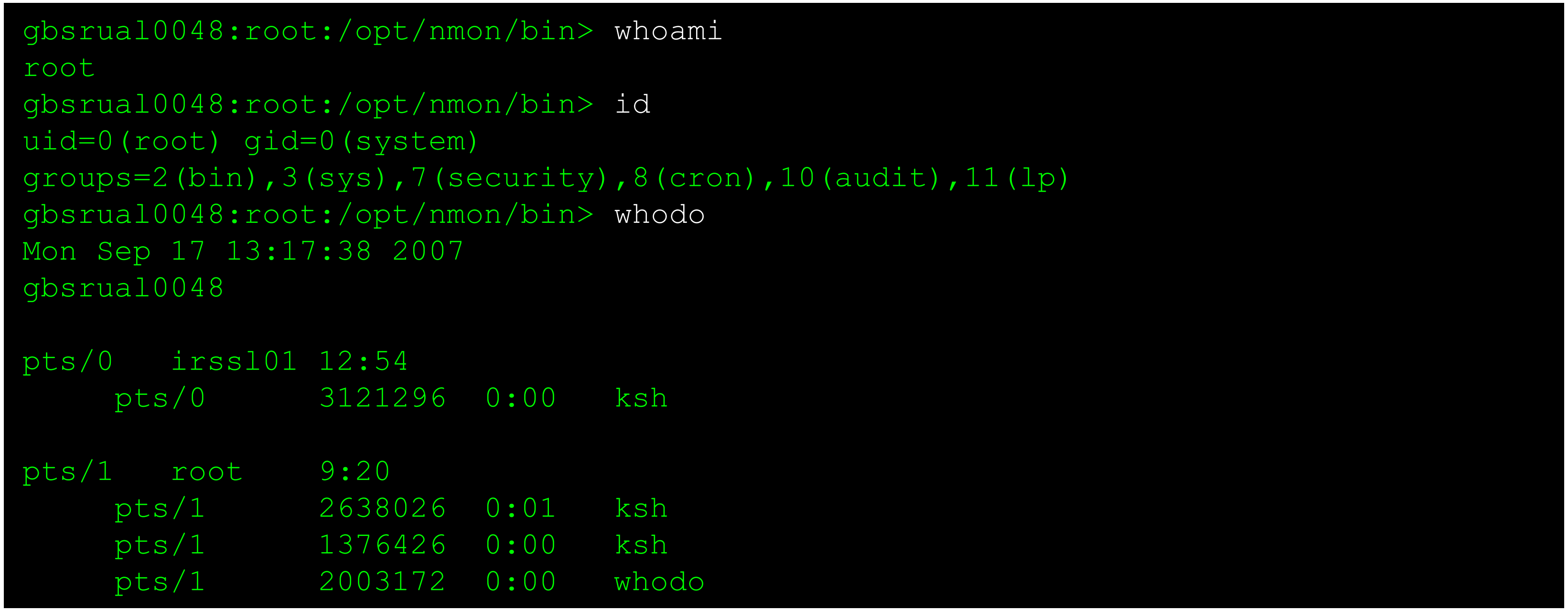
```
naac01:/home/hounselg> logout
```

UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ **Access to UNIX Systems**
  - **Logging on and off**
  - **whoami & whereami**
  - **Changing Users**
- □ Processes
- □ Filesystems & Directories
- □ Devices

whoami & whereami?

- ❑ Who Am I?
  - **Use the command whoami or id or whodo**



```
gbsrual0048:root:/opt/nmon/bin> whoami
root
gbsrual0048:root:/opt/nmon/bin> id
uid=0(root) gid=0(system)
groups=2(bin),3(sys),7(security),8(cron),10(audit),11(lp)
gbsrual0048:root:/opt/nmon/bin> whodo
Mon Sep 17 13:17:38 2007
gbsrual0048

pts/0   irssl01 12:54
     pts/0      3121296  0:00   ksh

pts/1   root    9:20
     pts/1      2638026  0:01   ksh
     pts/1      1376426  0:00   ksh
     pts/1      2003172  0:00   whodo
```

- ❑ Used to identify who you are logged in on the system and what they are doing

whoami & whereami?

❑ Where Am I?

- **Use the command pwd**
- **Used to identify where you are (which directory) on the system**

```
syhp79# pwd
/capacity/scripts/newscripts
syhp79#
```

❑ Use the hostname (or uname –a) to check which server you are logged onto.

```
gbsrual0048:root:/opt/nmon/bin> hostname
gbsrual0048
gbsrual0048:root:/opt/nmon/bin> uname -a
AIX gbsrual0048 3 5 0002A587D600
```

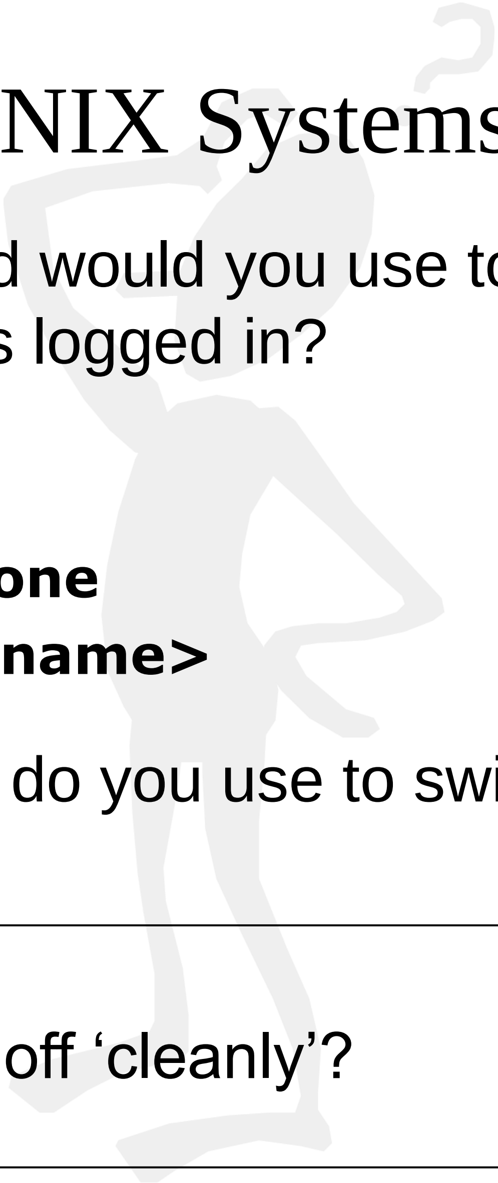
UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ Access to UNIX Systems
  - **Logging on and off**
  - **Whoami & whereami**
  - **Switching Users**
- □ Processes
- □ Filesystems & Directories
- □ Devices

Accessing UNIX Systems

- Why do we need to switch users?
  - **Switch User to become another system user, typically root (super user).**
  - **System user accounts cannot (typically) directly login to a system, so su is used to become this user.**
  - **Switching users allows us to user another users environment, files, directories, programs, shell scripts, etc.**
- The **su** command is used to change users

Accessing UNIX Systems Checkpoint



- □ Which command would you use to find out when a particular user is logged in?

  a) whoami

  b) who

  c) finger everyone

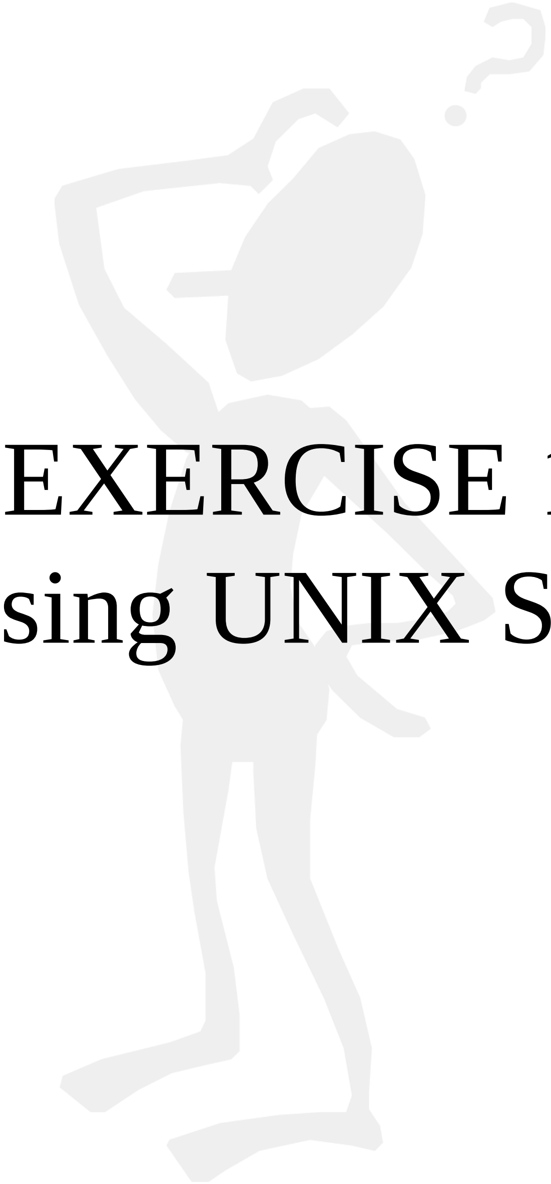
  d) finger <username>

- □ What command do you use to switch users?

  ____

- □ How do you log off ‘cleanly’?

  ____



EXERCISE 1
Accessing UNIX Systems

EXERCISE 2
UNIX Commands & Shell Basics

UNIX Fundamentals Part I

- ❑ UNIX History
- ❑ The Many Flavours' of UNIX
- ❑ The Structure of UNIX
- ❑ Access to UNIX Systems
- ❑ **Processes**
- ❑ Filesystems & Directories
- ❑ Devices

Processes

- Operating systems are about managing resources
  - **Files are about storage (and I/O) resources**
  - **Processes are about CPU resources**
- All programs that run on UNIX are termed “processes”.
- A Definition : *“A process is a single program running in its own virtual address space, and it receives a share (or time slice) of the CPU”*
- Processes are identified by their process identifier, an integer.
- Processes and commands are not the same
  - **A simple command (e.g. ls) generates a single process**
  - **complex commands or scripts can invoke several simultaneous executing processes**
- There are three types of processes in UNIX
  - **Interactive**
  - **daemons**
  - **Batch**

UNIX Fundamentals Part I

- UNIX History
- The Many Flavours' of UNIX
- The Structure of UNIX
- Access to UNIX Systems
- Processes
  - **Process Types**
  - **Job Control For Processes**
  - **Process Lifecycle**
  - **Process Control**
  - **Process Attributes**
- Filesystems & Directories
- Devices

Process Types - I

- Interactive processes:
  - **initiated and controlled by terminal session**
  - **can accept input from user as it runs**
  - **can output results to the terminal**
  - **Unix has job control to manage processes**

Process Types - II

- ❑ Some processes run continuously listening for input, these are normally called “daemons”.
- ❑ daemons are
  - **server process running in the background**
  - **often started at boot time**
  - **offer a service to other processes**
- ❑ examples
  - **ftpd - file transfer process daemon**
  - **cron daemon - scheduling daemon**
  - **httpd - web server**

Process Types - III

❑ Batch processes are

- **not associated with any terminal**
- **jobs that are submitted to a queue to await scheduling**
- **only basic support in Unix by default compared with other mainframe op sys.**
- **3rd part system such as NASA's Network Queue System can be used**

UNIX Fundamentals Part I

- ❑ UNIX History
- ❑ The Many Flavours' of UNIX
- ❑ The Structure of UNIX
- ❑ Access to UNIX Systems
- ❑ Processes
  - Process Types
  - **Job Control For Processes**
  - Process Lifecycle
  - Process Control
  - Process Attributes
- ❑ Filesystems & Directories
- ❑ Devices

Job Control For Processes - I

- foreground process
  - **by default interactive processes run in the foreground and the shell must wait until they complete**
  - **only one process can be running in the foreground for each user.**
  - **But Remember that Unix is a multi-user system - foreground and background relate to user sessions. Hence multiple 'foreground' processes can be running.**

Job Control For Processes - II

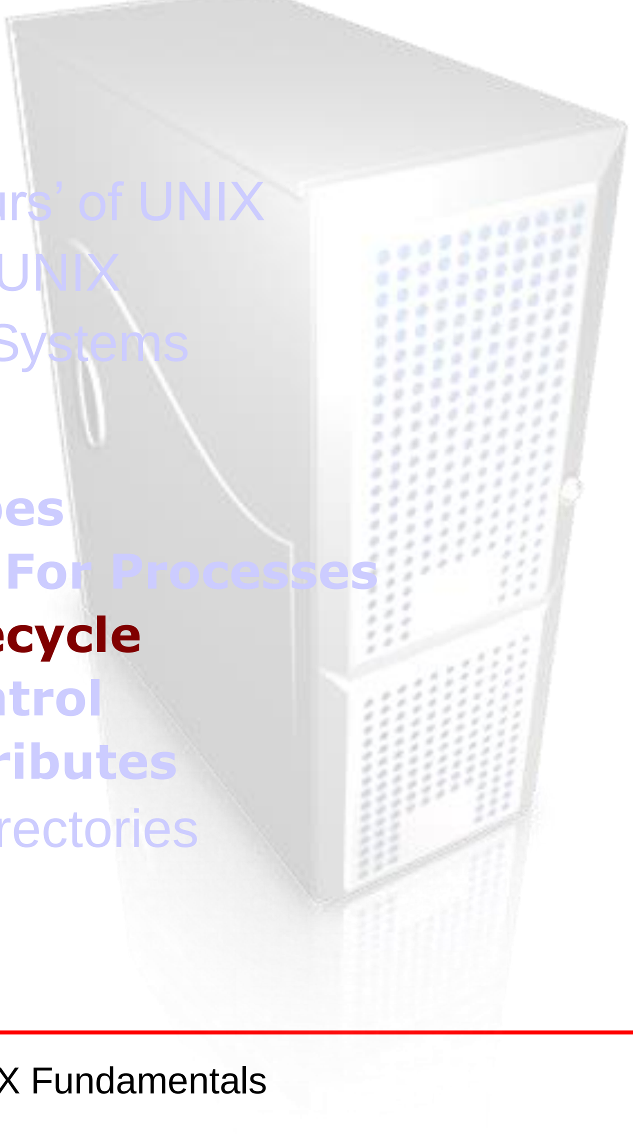
- Background process
  - **if a process has no output to terminal and will take some time to run, rather than waiting, it can be run as a background process**
  - **Once started in the background control returns immediately to the shell.**
  - **a user can initiate multiple simultaneous background processes**
  - **with the bash shell following a command with an & places in the background**

    eg makewhatis &

Job Control For Processes - III

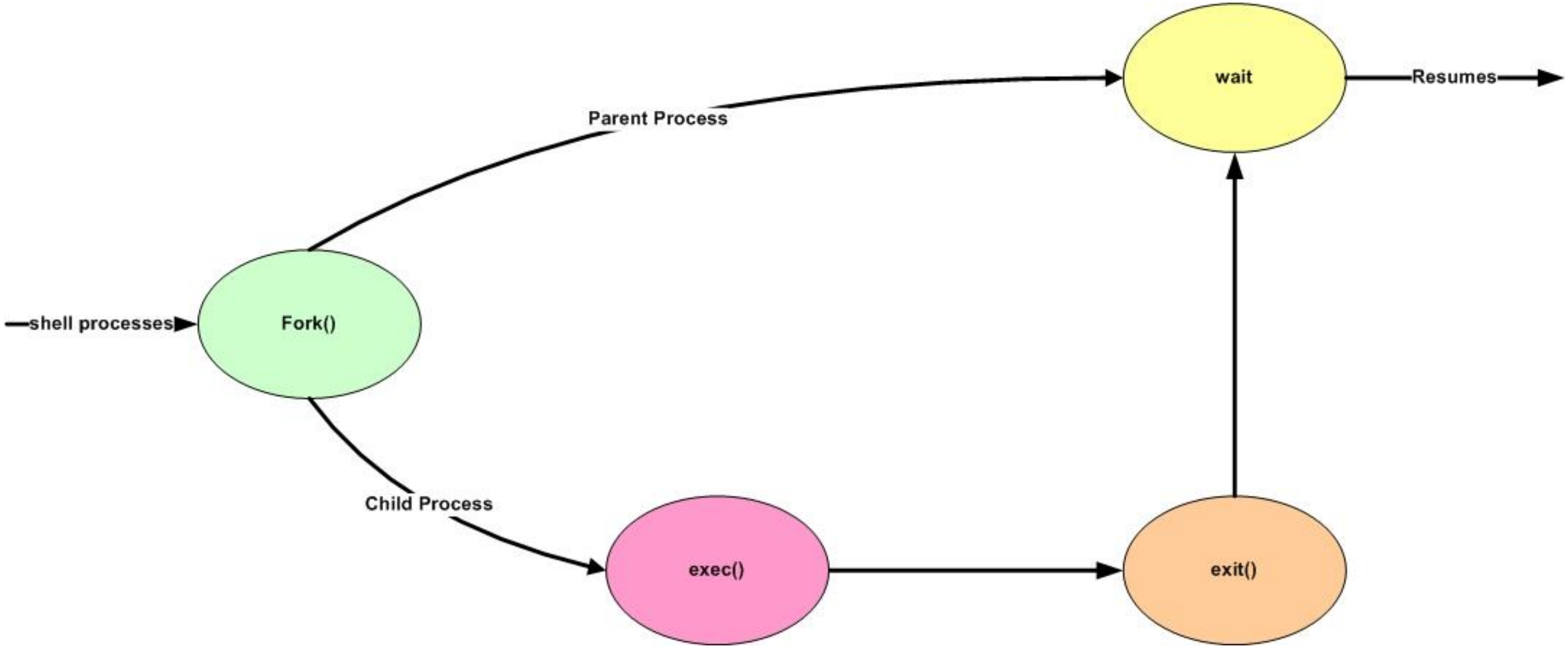
- The ‘jobs’ command show what process are suspended or running in the background
- Moving a process from the foreground to the background takes two steps
- A process running in the foreground can be suspended (ctrl-Z)
- A suspended process can be placed in the background (use bg command)
- One suspended or background process can be brought to the foreground (fg command)

UNIX Fundamentals Part I

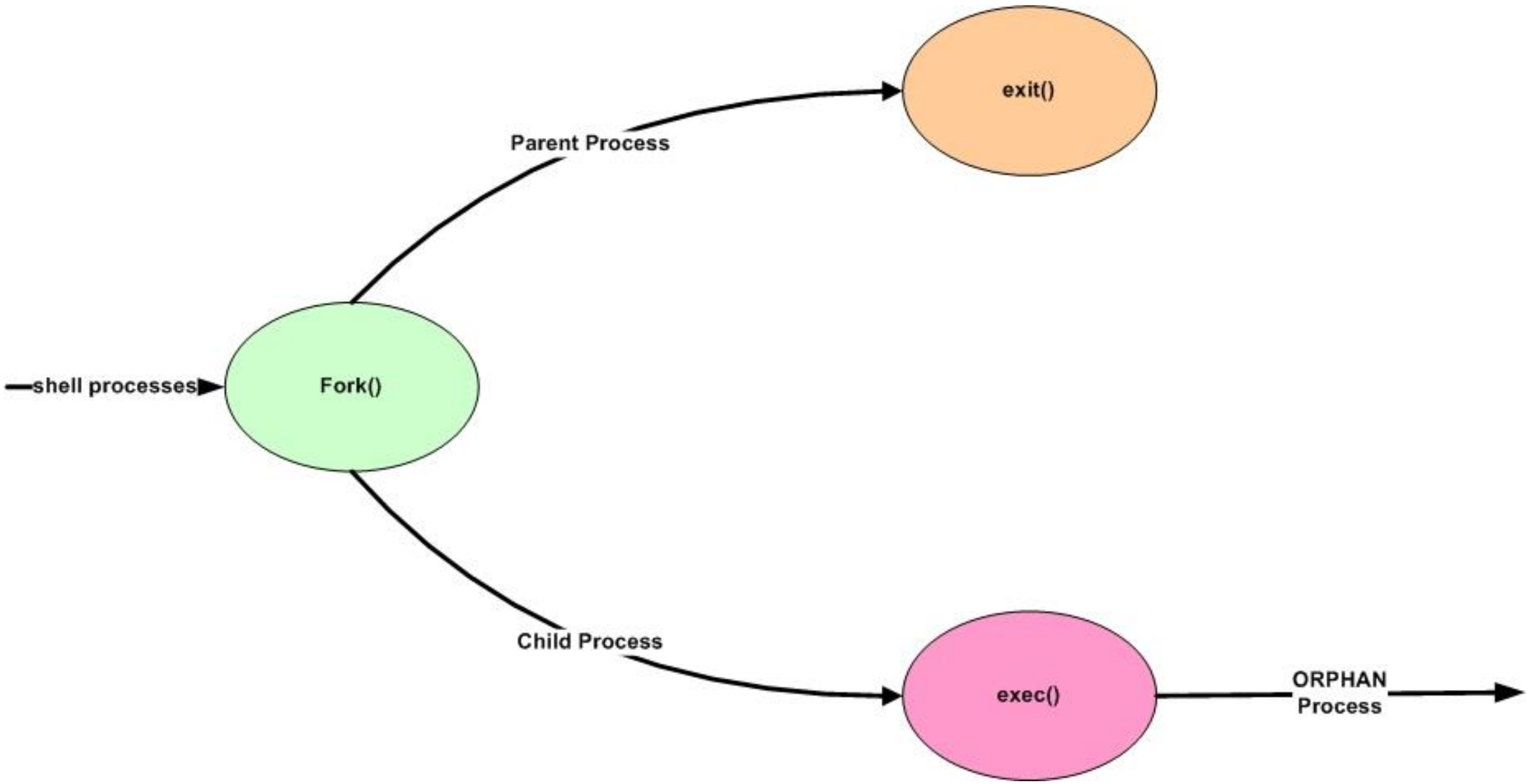


- ❑ UNIX History
- ❑ The Many Flavours' of UNIX
- ❑ The Structure of UNIX
- ❑ Access to UNIX Systems
- ❑ **Processes**
  - Process Types
  - Job Control For Processes
  - **Process Lifecycle**
  - Process Control
  - Process Attributes
- ❑ Filesystems & Directories
- ❑ Devices

Process Lifecycle - I

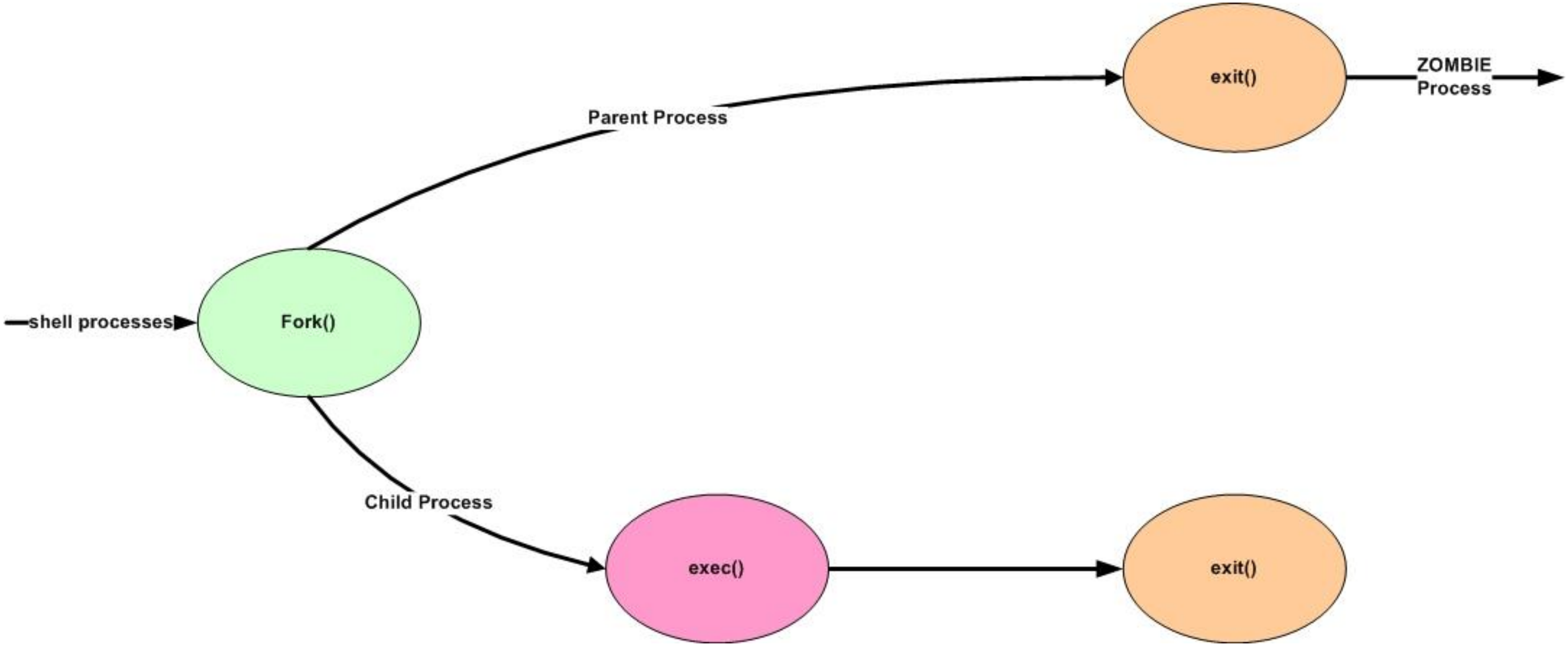


Process Lifecycle - II



- **Orphan Processes** occur when a parent process dies without first killing its child processes which become orphans.

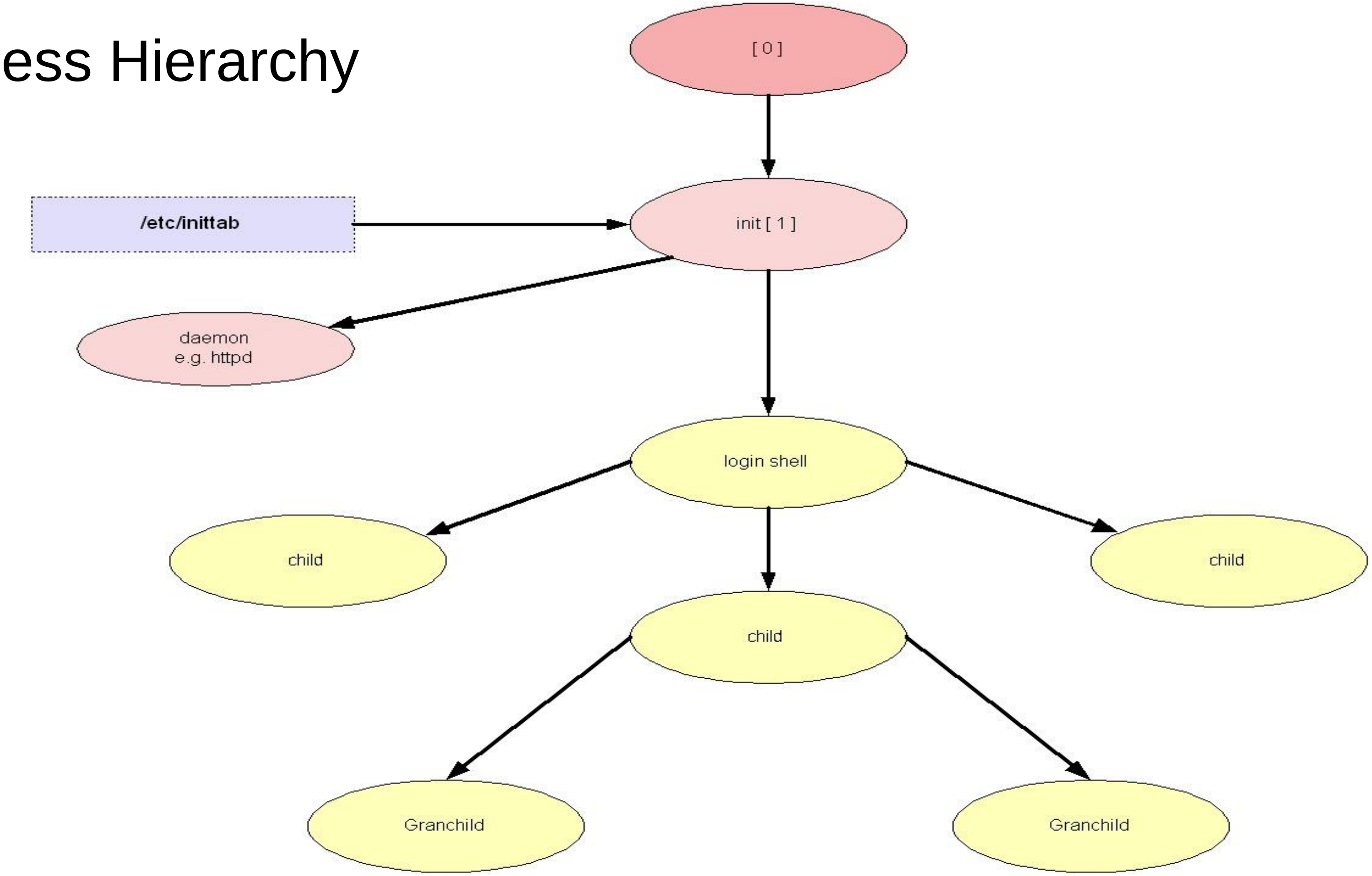
Process Lifecycle - III



- A **zombie process** results when the parent of a defunct child process exits before the terminated child.

Process Lifecycle - IV

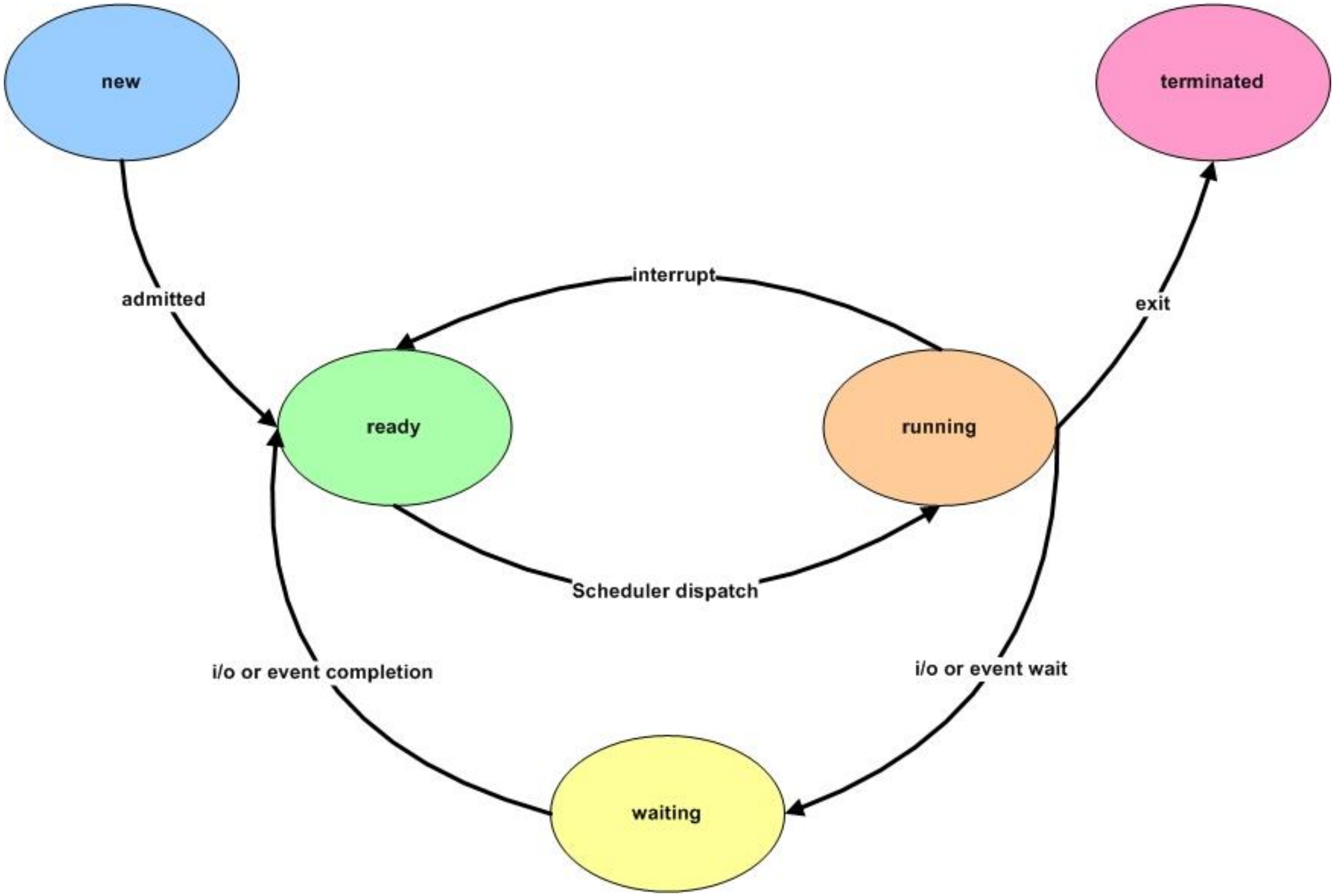
Process Hierarchy



UNIX Fundamentals Part I

- UNIX History
- The Many Flavours' of UNIX
- The Structure of UNIX
- Access to UNIX Systems
- **Processes**
  - Process Types
  - Job Control For Processes
  - Process Lifecycle
  - **Process Control**
  - Process Attributes
- Filesystems & Directories
- Devices

Process Control - I



Process Control – II Signals

- ❑ Facility for handling exceptional conditions similar to software interrupts (*kill –l* shows signal list)
- ❑ The *interrupt* signal, SIGINT, is used to stop a command before that command completes (usually produced by ^C).
- ❑ Signal use has expanded beyond dealing with exceptional events.
  - **Start and stop subprocesses on demand**
  - **SIGWINCH informs a process that the window in which output is being displayed has changed size.**
  - **Deliver urgent data from network connections.**

Process Control - III

- setuid bit sets the effective user identifier of the process to the user identifier of the owner of the file, and leaves the real user identifier as it was.

- setuid scheme allows certain processes to have more than ordinary privileges while still being executable by ordinary users.

UNIX Fundamentals Part I

- UNIX History
- The Many Flavours' of UNIX
- The Structure of UNIX
- Access to UNIX Systems
- **Processes**
  - Process Types
  - Job Control For Processes
  - Process Lifecycle
  - Process Control
  - **Process Attributes**
- Filesystems & Directories
- Devices

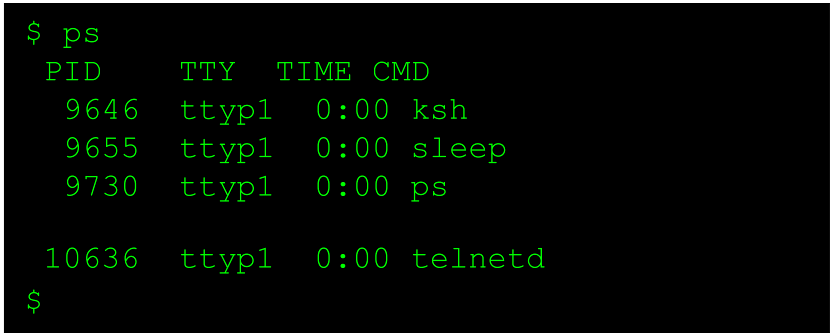
Process Attributes - 1

- the ps and top commands can be used to look at current processes (i.e. to view the process table).
  - **PID - process ID : each process has a unique ID**
  - **PPID parent process ID : The process that ‘forked’ to start a process**
  - **nice value - priority (-20 highest to 19 = lowest)**
  - **TTY associated terminal (TTY teletype terminal)**

- The process table is a snapshot of all processes running at that time.

Process Attributes - 2

- ❑ The “ps” command on its own will show you your own processes.
- ❑ Each process has a process id (pid) that uniquely identifies it.
- ❑ We can also see the tty device the process was started on and how long it has been running.



```
$ ps
 PID    TTY  TIME CMD
  9646  ttyp1  0:00 ksh
  9655  ttyp1  0:00 sleep
  9730  ttyp1  0:00 ps

 10636  ttyp1  0:00 telnetd
$
```

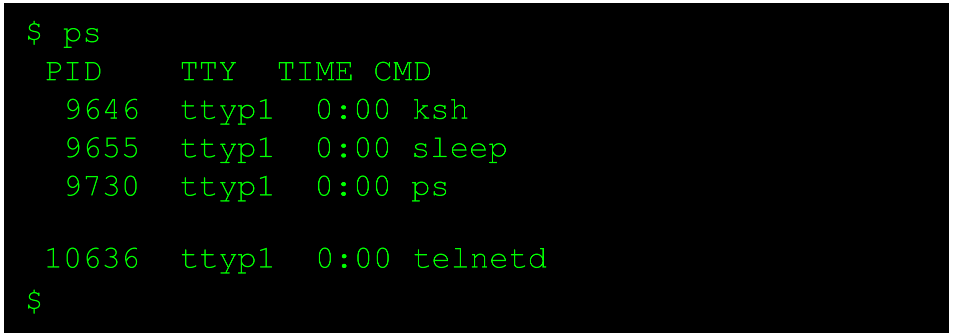
Process Attributes - 3

- This process table shows the ps command itself, the shell used to invoke it and a telnetd process.

- When you log in, you automatically start a shell.

- To see what the PID of the current shell (a running process) type:

  echo $$

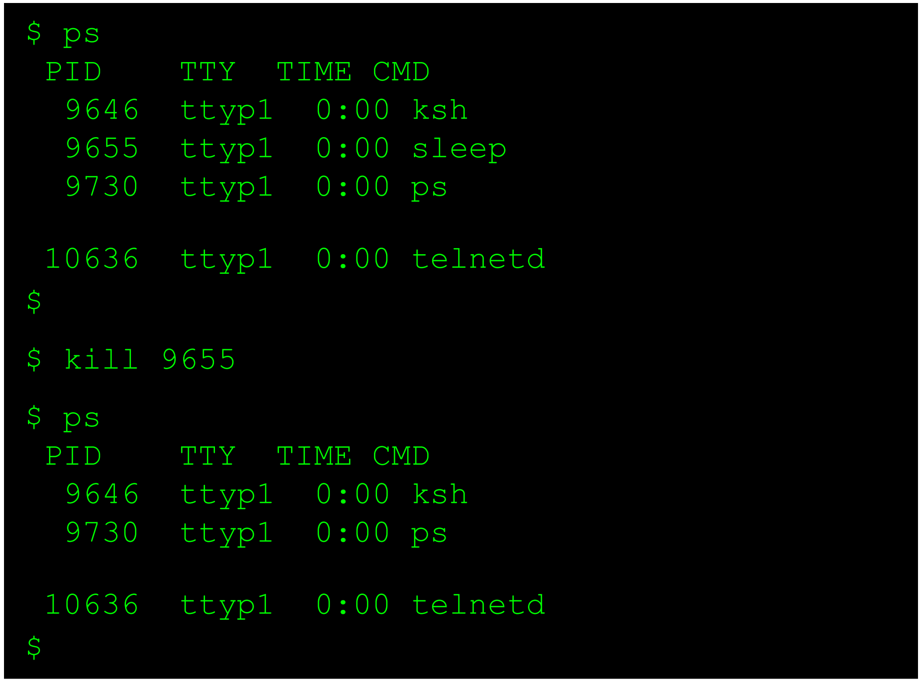


```
$ ps
 PID     TTY   TIME CMD
  9646  ttyp1   0:00 ksh
  9655  ttyp1   0:00 sleep
  9730  ttyp1   0:00 ps

 10636  ttyp1   0:00 telnetd
$
```

Process Attributes - 4

- We can kill a process (providing we have permission to do so) using the PID.

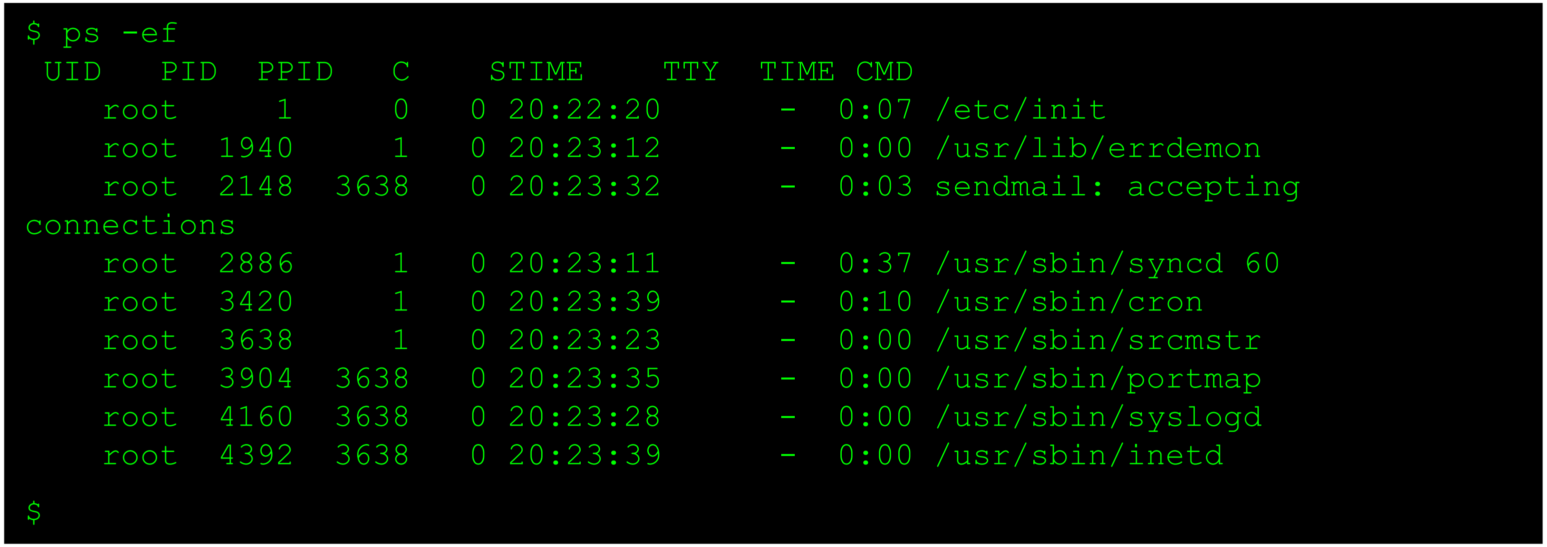
- The kill command is used to stop the sleep process.



```
$ ps
 PID     TTY   TIME CMD
  9646  ttyp1   0:00 ksh
  9655  ttyp1   0:00 sleep
  9730  ttyp1   0:00 ps

 10636  ttyp1   0:00 telnetd
$

$ kill 9655

$ ps
 PID     TTY   TIME CMD
  9646  ttyp1   0:00 ksh
  9730  ttyp1   0:00 ps

 10636  ttyp1   0:00 telnetd
$
```

Process Attributes - 5

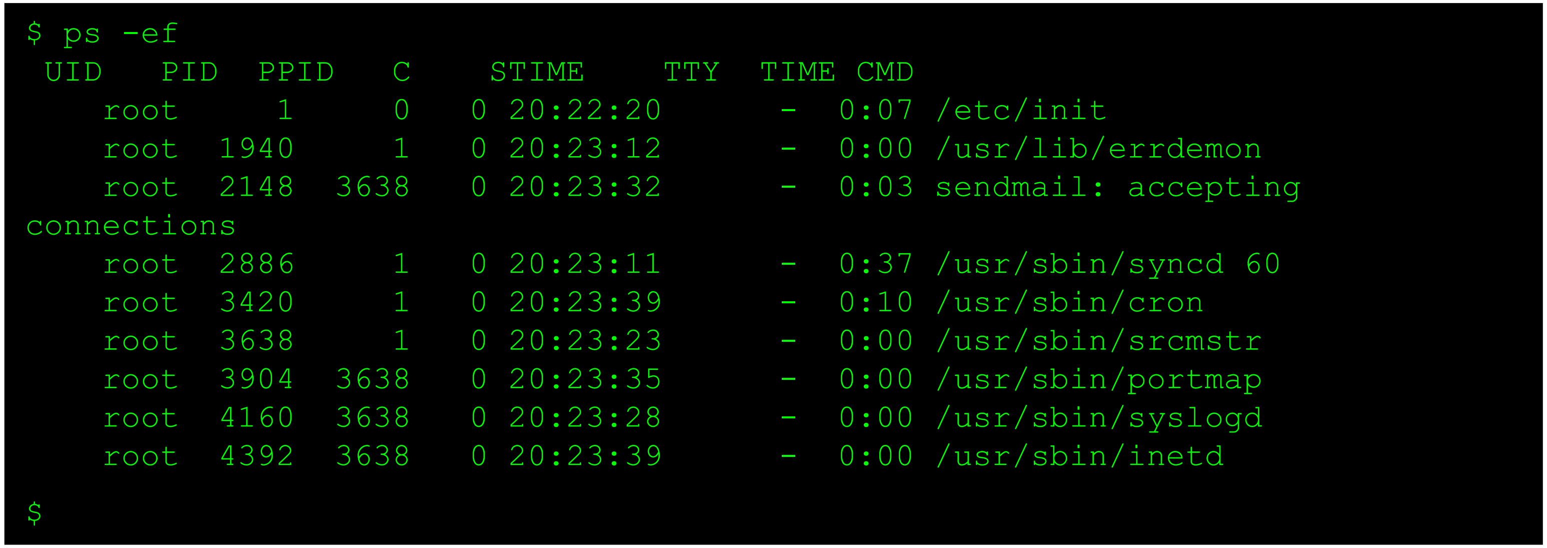
- ❑Using the options “-ef” with the “ps” command shows all the processes running on the computer.
- ❑**Any** user is allowed to do this.



```
$ ps -ef
 UID   PID  PPID   C    STIME    TTY  TIME CMD
    root     1     0   0 20:22:20      -  0:07 /etc/init
    root  1940     1   0 20:23:12      -  0:00 /usr/lib/errdemon
    root  2148  3638   0 20:23:32      -  0:03 sendmail: accepting
connections
    root  2886     1   0 20:23:11      -  0:37 /usr/sbin/syncd 60
    root  3420     1   0 20:23:39      -  0:10 /usr/sbin/cron
    root  3638     1   0 20:23:23      -  0:00 /usr/sbin/srcmstr
    root  3904  3638   0 20:23:35      -  0:00 /usr/sbin/portmap
    root  4160  3638   0 20:23:28      -  0:00 /usr/sbin/syslogd
    root  4392  3638   0 20:23:39      -  0:00 /usr/sbin/inetd

$
```

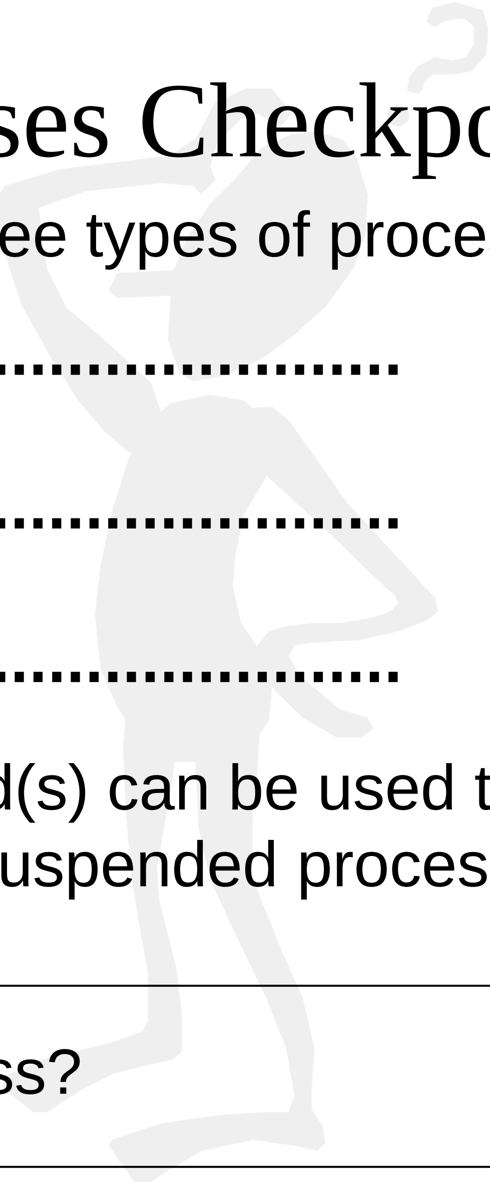
Process Attributes - 6

- Each process shows its owner (UID), the process ID (PID), the parent process ID (PPID), CPU utilisation, start time, the tty is was started on, and the total execution time.



```
$ ps -ef
 UID    PID  PPID    C     STIME     TTY  TIME CMD
    root      1     0    0 20:22:20       -  0:07 /etc/init
    root   1940     1    0 20:23:12       -  0:00 /usr/lib/errdemon
    root   2148  3638    0 20:23:32       -  0:03 sendmail: accepting
connections
    root   2886     1    0 20:23:11       -  0:37 /usr/sbin/syncd 60
    root   3420     1    0 20:23:39       -  0:10 /usr/sbin/cron
    root   3638     1    0 20:23:23       -  0:00 /usr/sbin/srcmstr
    root   3904  3638    0 20:23:35       -  0:00 /usr/sbin/portmap
    root   4160  3638    0 20:23:28       -  0:00 /usr/sbin/syslogd
    root   4392  3638    0 20:23:39       -  0:00 /usr/sbin/inetd

$
```

Processes Checkpoint (1)



- □ What are the three types of process?

  1. ....

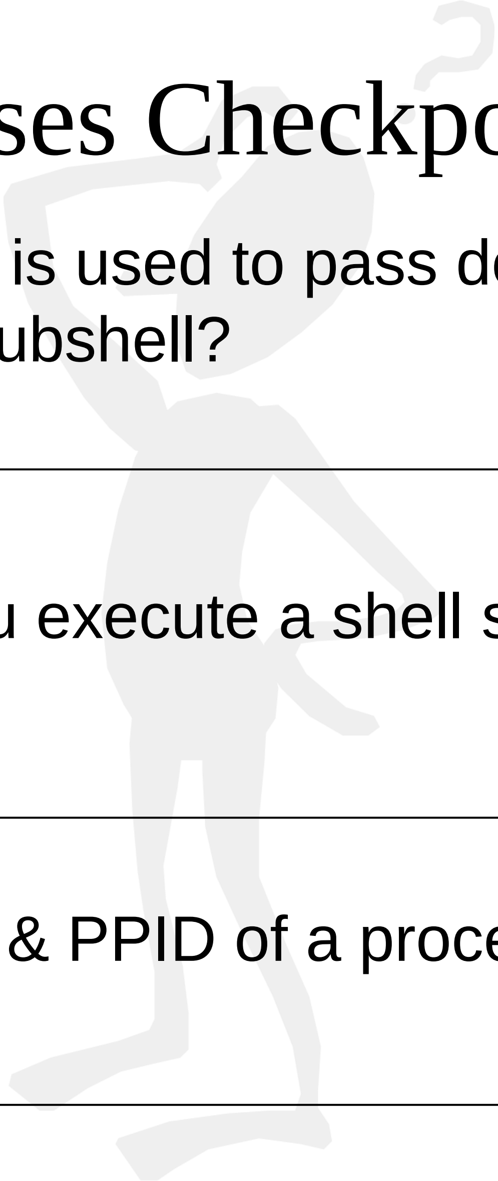
  2. ....

  3. ....

- □ Which command(s) can be used to check for background or suspended processes?

- □ What is a process?

Processes Checkpoint (2)



□ What command is used to pass down the value of a variable into a subshell?

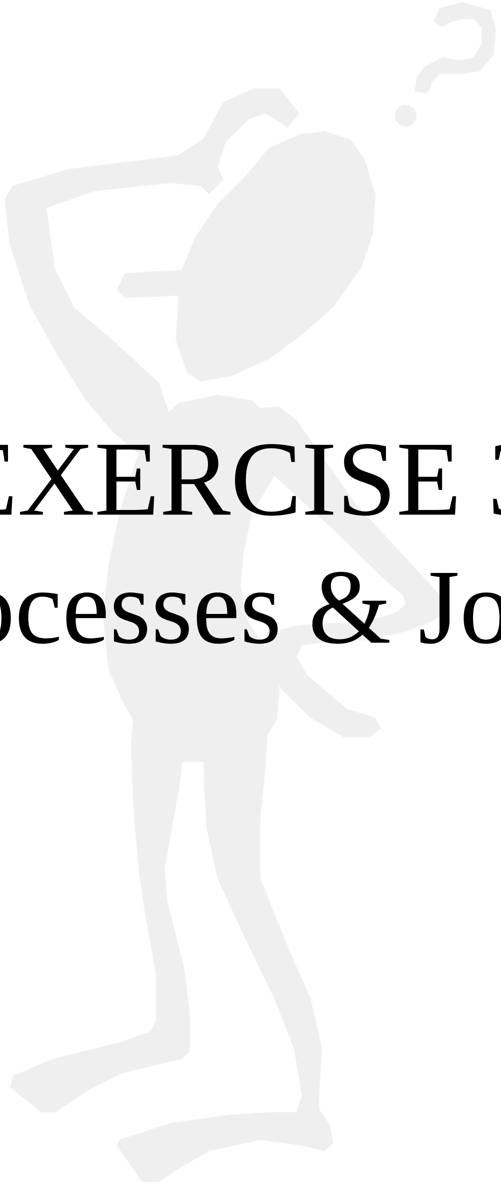
____

□ When would you execute a shell script using the (.) dot notation?

____

□ What is the PID & PPID of a process?

____



EXERCISE 3
UNIX Processes & Job Control

UNIX Fundamentals Part I

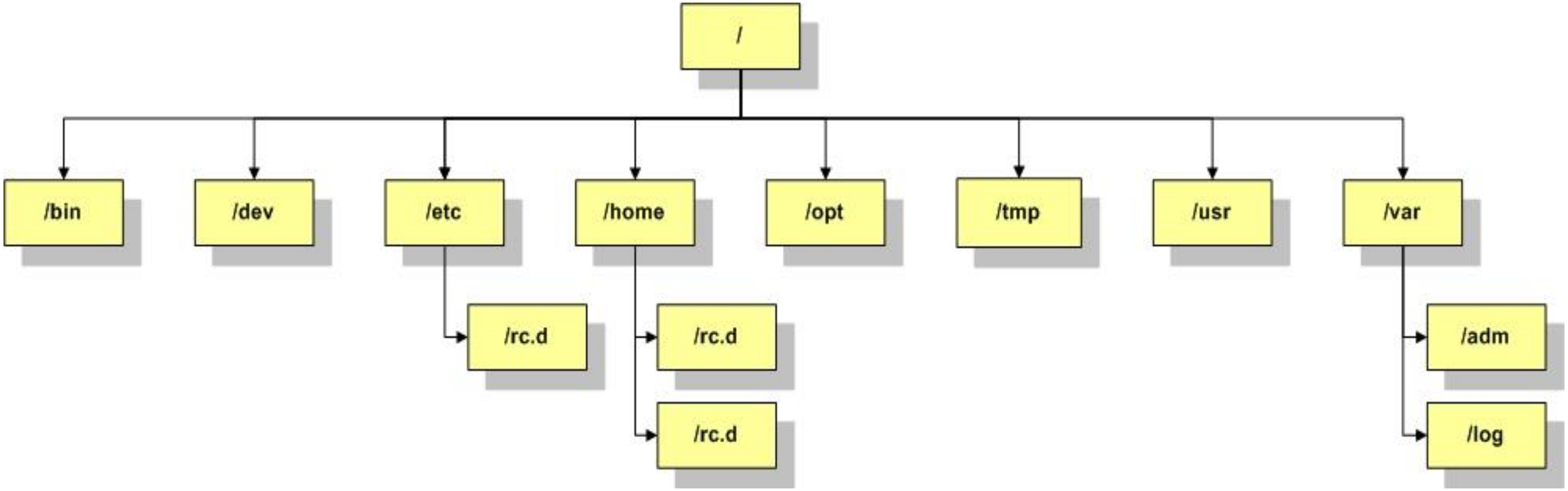
- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ Access to UNIX Systems
- □ Processes
- □ **Filesystems & Directories**
- □ Devices

Filesystems & Directories

- ❑ Directory Structures
- ❑ Permissions/File Access Modes
- ❑ Directory & File Commands

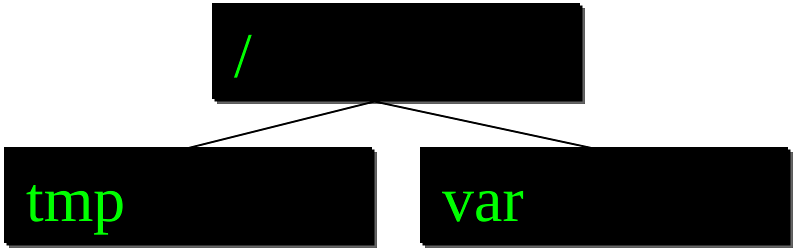
Directory Structures

- ❑ Directories within UNIX are hierarchical and start with the root directory or “/”.
- ❑ All other directories are underneath the root directory.
- ❑ Every directory has a parent, and possibly one or more children. Children can in turn be parents.
- ❑ A directory is a special type of file (so a file and a directory of the same name within the same directory is impossible).



Directory Structures

- ❑ In this example the directories “tmp” and “var” are shown under “/”.

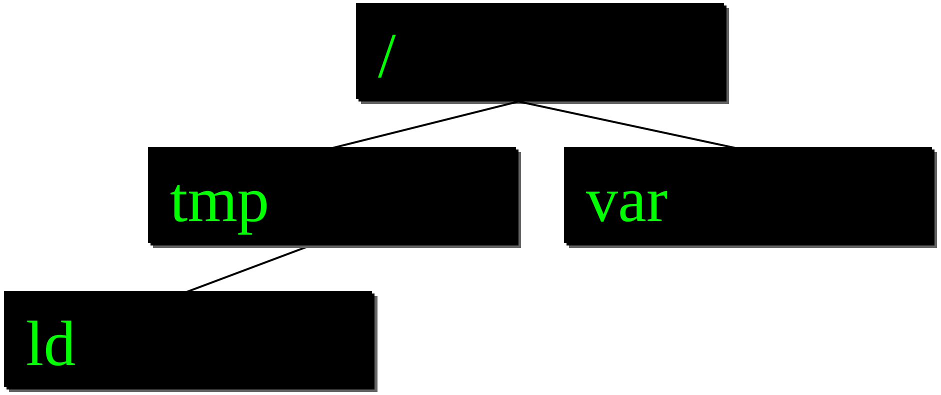


- ❑ These directories are referenced as “/tmp” and “/var”.

Directory Structures

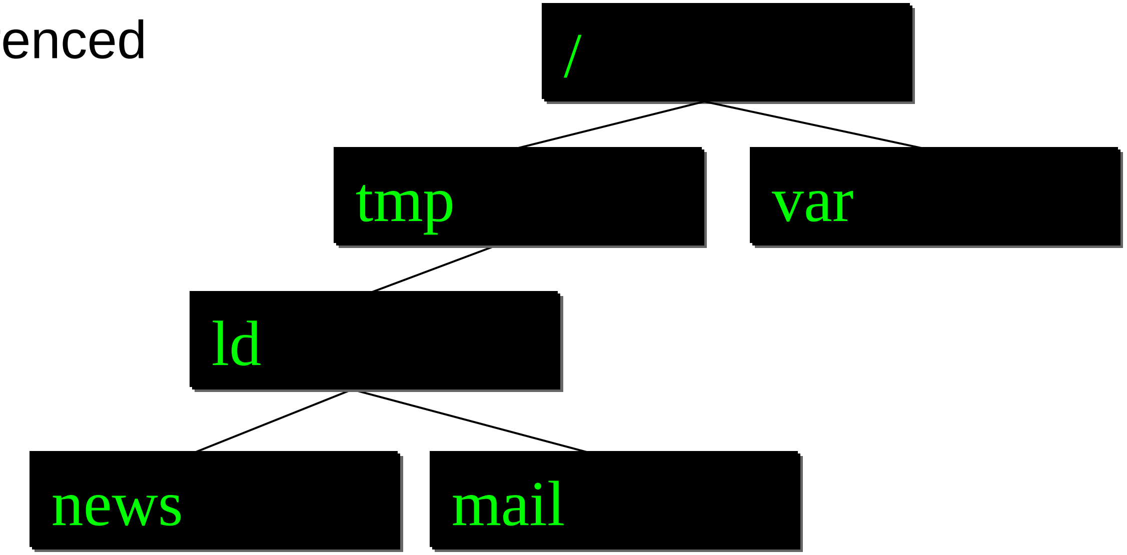
- A directory is created within the “/tmp” directory called “ld”.

- This is referenced as “/tmp/ld”.



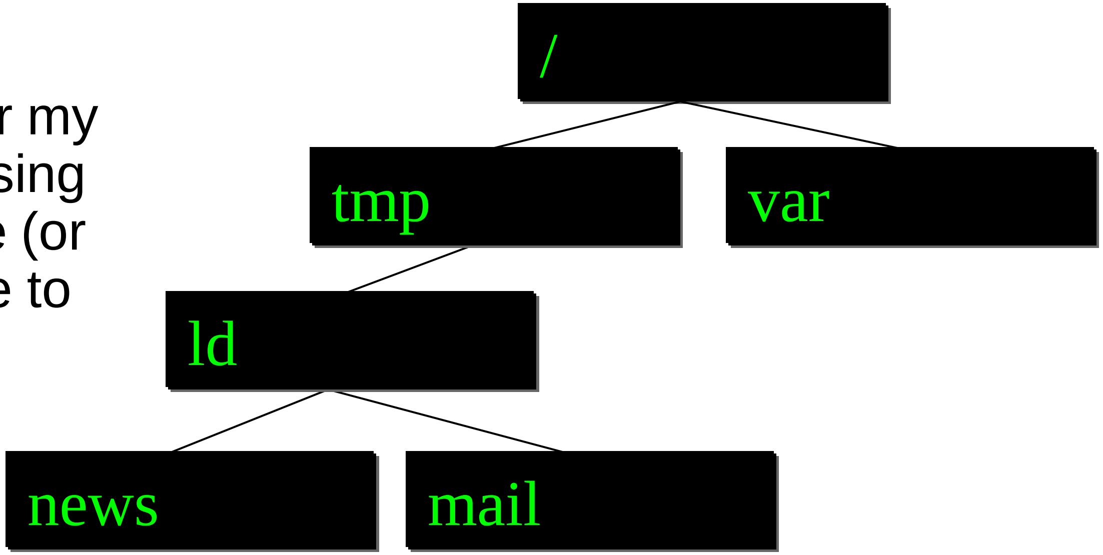
Directory Structures

- ❑ Two new directories are created under "/tmp/ld" called "news" and "mail".
- ❑ What are these referenced as?



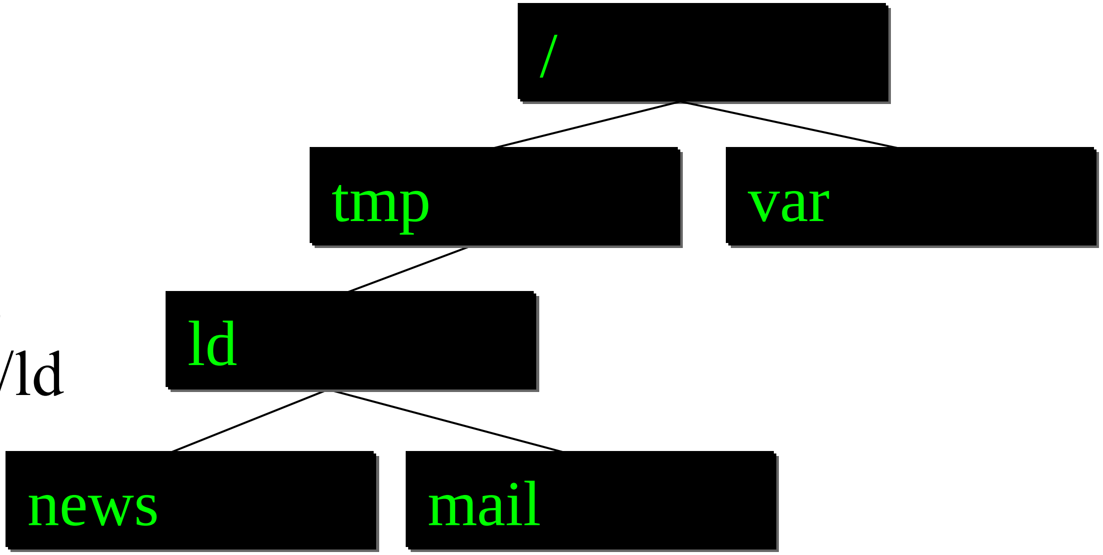
Directory Structures

- There are two ways to reference a directory; *relative* or *absolute*.
- So far we have seen *absolute* references.
- This means, whatever my current directory is, using an absolute reference (or pathname) will get me to that directory.



Directory Structures

- ❑ With relative pathnames the reference used is relative to the current directory.



❑If your in the /tmp directory you can use just “ld” to reference the /tmp/ld directory.

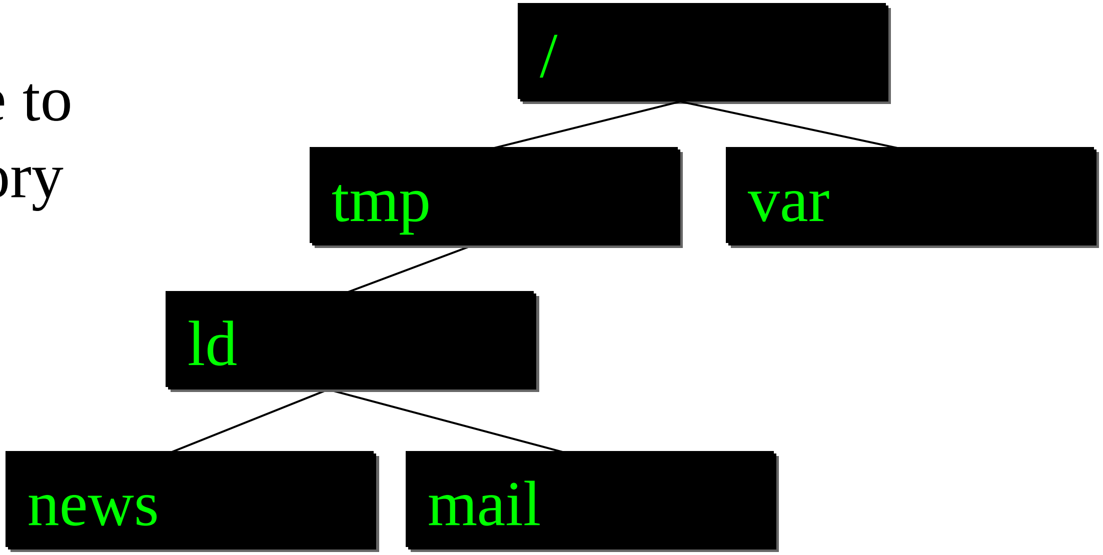
❑You can use cd “ld/news” to get to /tmp/ld/news IF you are already in /tmp.

Directory Structures

- ❑ Case sensitive. E.g. file1.tmp is a different directory (or file) to File1.tmp.
- ❑ Large filename lengths. E.g. `have_you_seen_the_size.of.this_filename.doc` is a valid directory name (or filename).
- ❑ Naming Conventions
  - ▪ **Simple, any ASCII characters can be used**
  - ▪ **It is recommended that <u>no</u> 'metacharacters' are used.**
  - ▪ **It is recommended to use only**
    - ***Letters***
    - ***Digits***
    - ***Underscore (_)***
    - ***Hyphen (-)***
    - ***Dot (.)***
- ❑ Filenames/Directories beginning with a dot (**.**) are hidden.

Directory Structures

- ❑ To move between these directories use the UNIX command “cd”.



❑For example, to move to the /tmp/ld/news directory use:

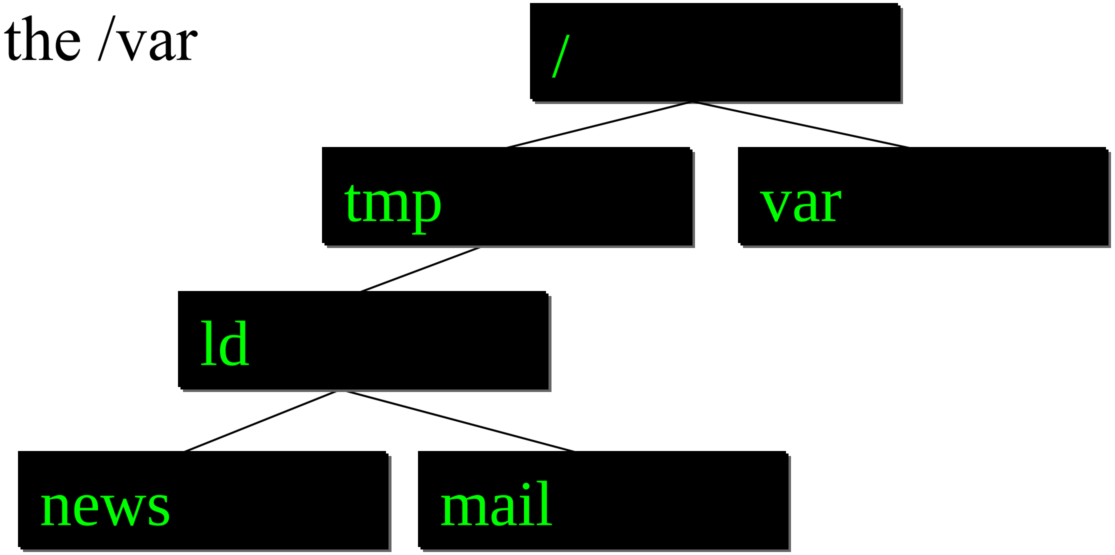
cd /tmp/ld/news

❑If I’m already in /tmp/ld, I can use “cd news”.

Directory Structures

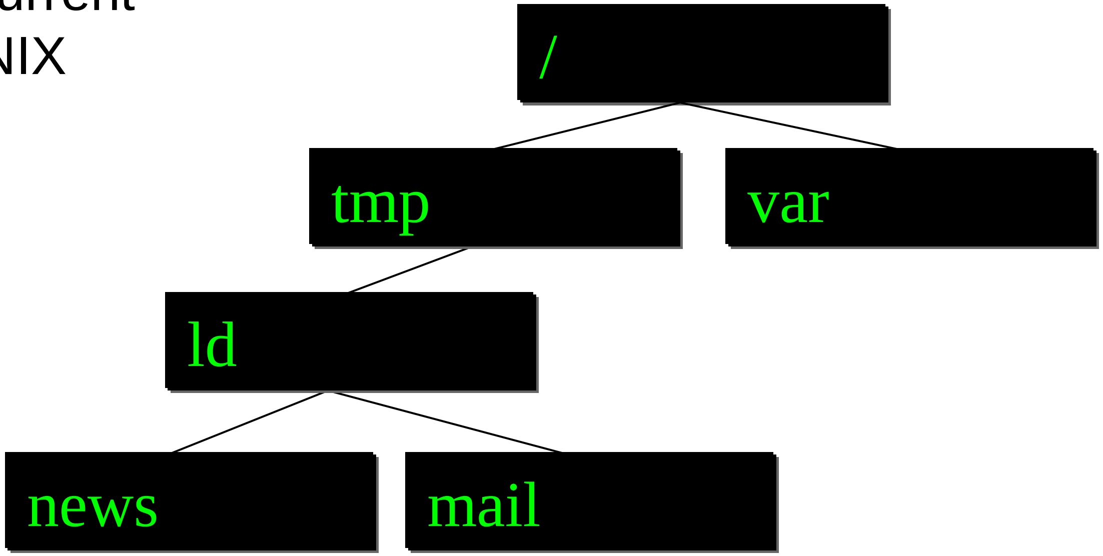
- ❑ use “../” to go back a directory to /.

- ❑use “../var” to get to the /var directory.



Directory Structures

- To find out what your current directory is, use the UNIX command “pwd”.



/

tmp

var

ld

news

mail

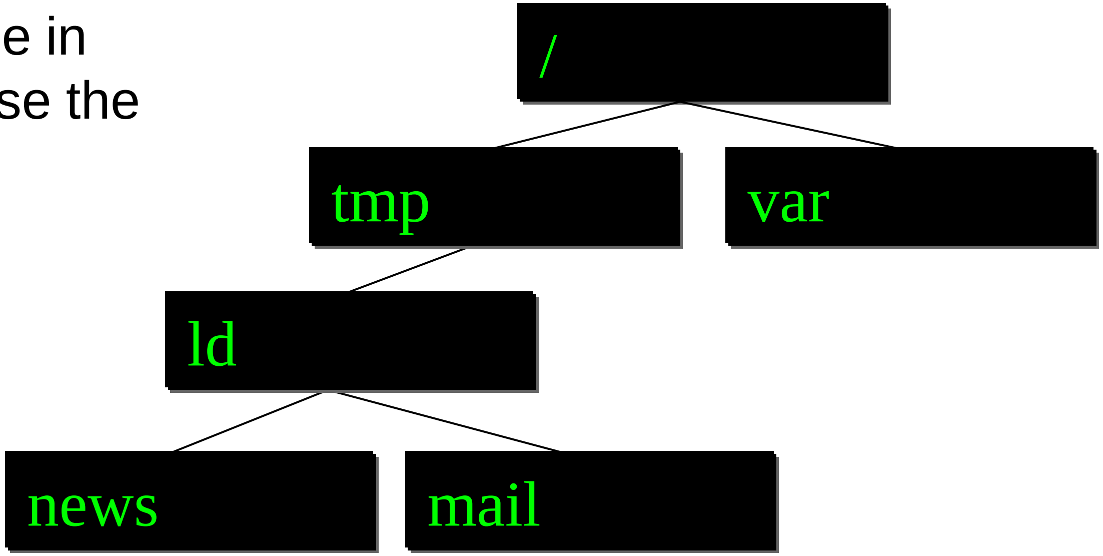
```
$ pwd
/tmp/ld
$
```

UNIX Fundamentals Part I

- ❑ UNIX History
- ❑ The Many Flavours' of UNIX
- ❑ The Structure of UNIX
- ❑ Access to UNIX Systems
- ❑ Processes
- ❑Filesystems & Directories
  - ▪ **Directory Structures**
  - ▪ **Permissions/File Access Modes**
  - ▪ **Directory & File Commands**
- ❑ Devices

Permissions/File Access Modes

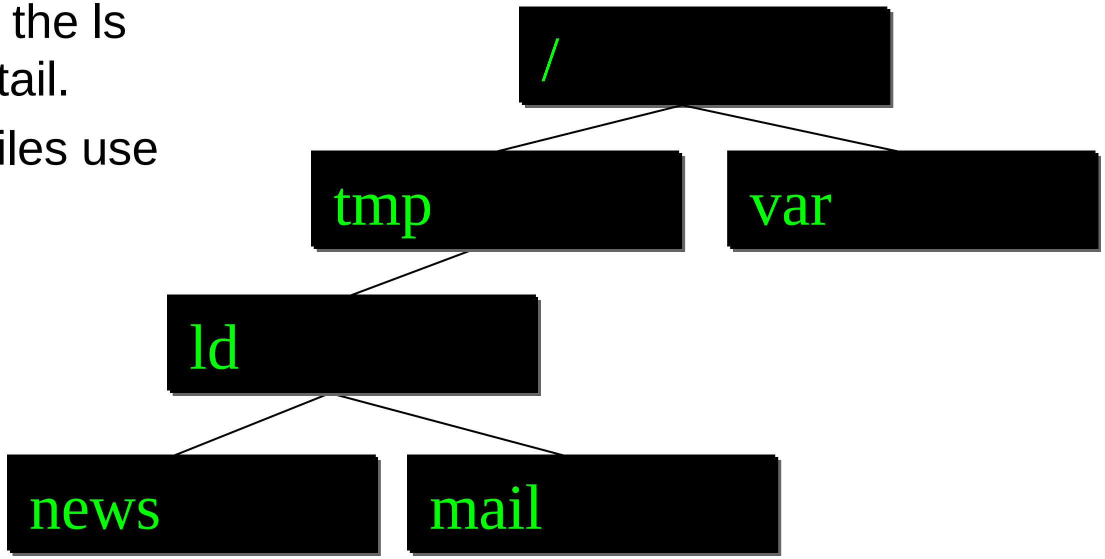
- ❑ To see what files reside in the current directory use the UNIX command “ls”.



```
$ ls
file1.txt
$
```

Permissions/File Access Modes

- ❑ Other options used with the ls command give more detail.
- ❑ To see a long listing of files use “ls -l”.



/

tmp

var

ld

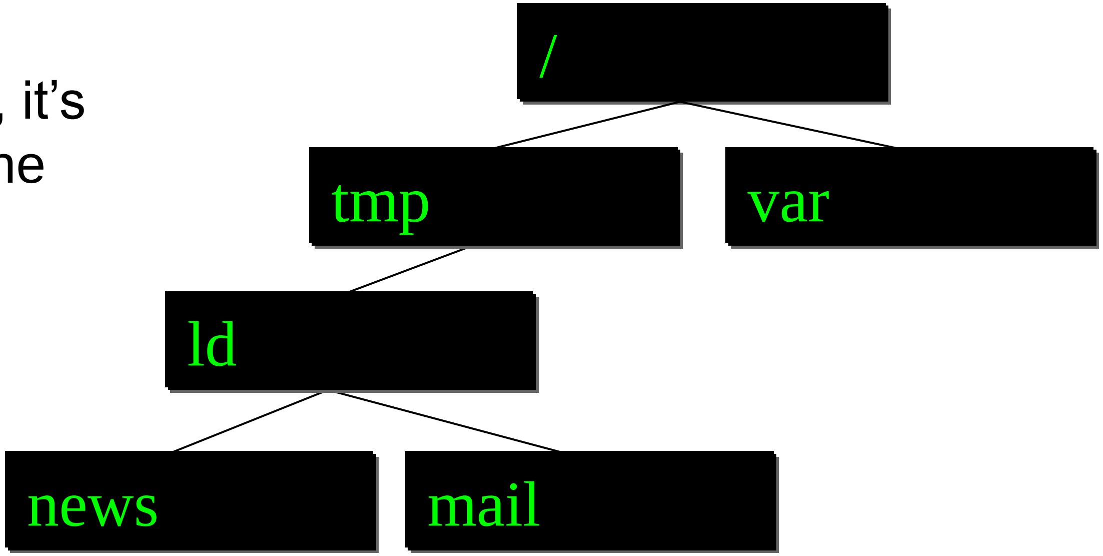
news

mail

```
$ ls -l
-rw-rw-rw- 1 root   sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

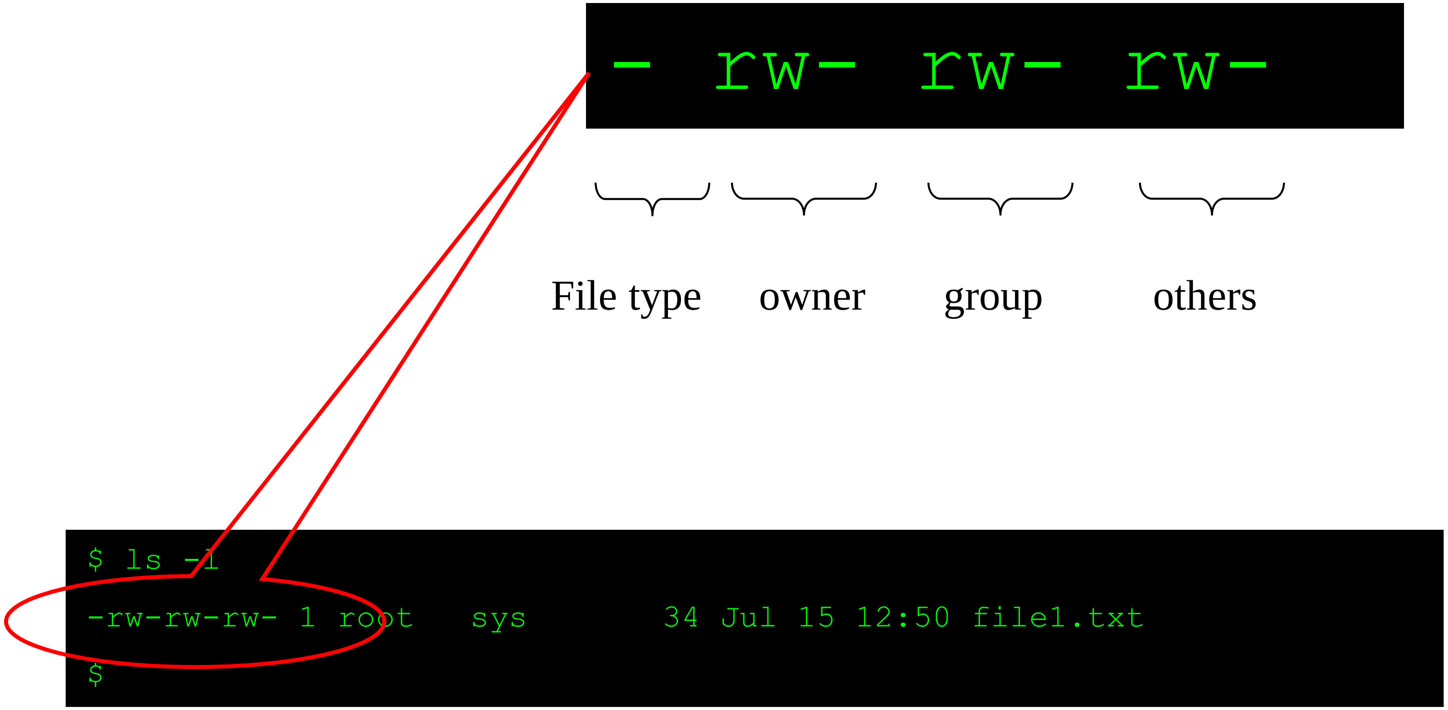
- ❑ Now we can see the permissions of the file, it’s owner, its group and the date/time it was last modified.



```
$ ls -l
-rw-rw-rw- 1 root   sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

- ❑ The permissions consist of...



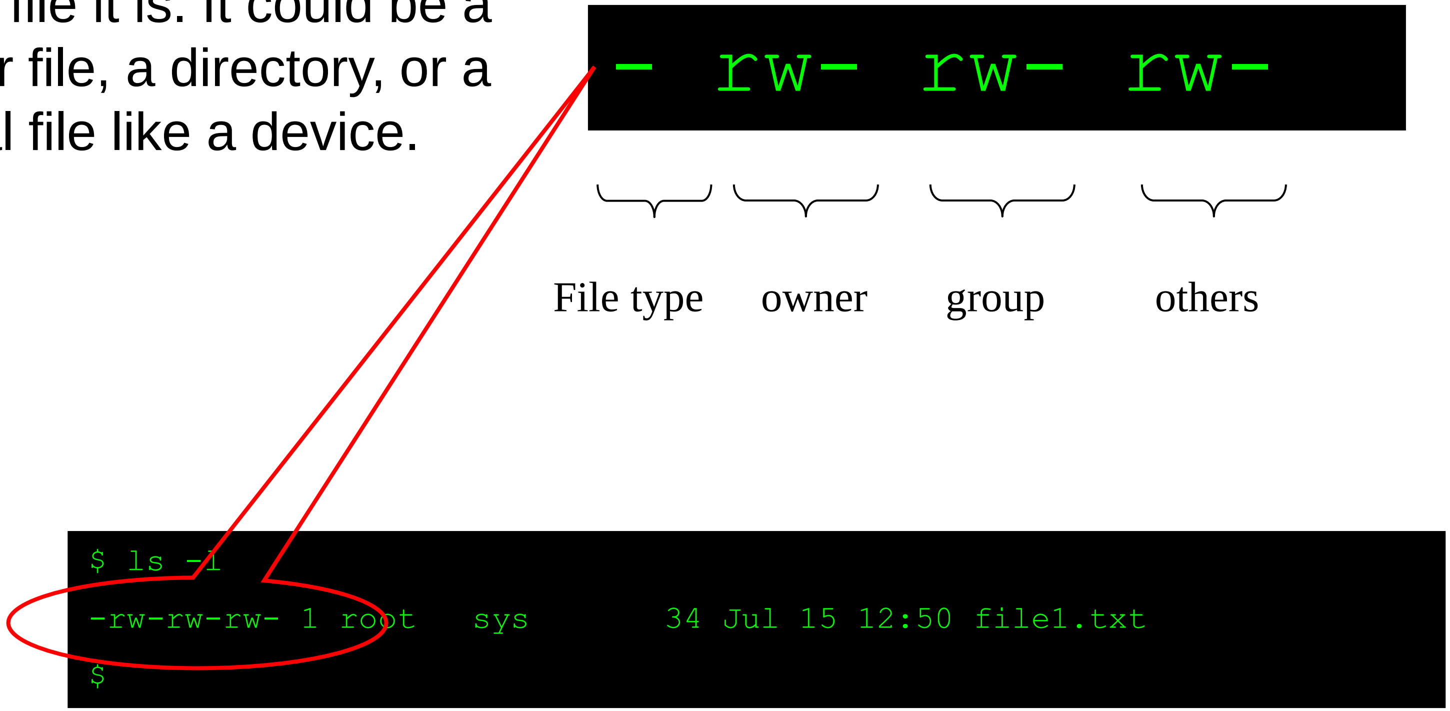
```
- rw- rw- rw-
```

File type owner group others

```
$ ls -l
-rw-rw-rw- 1 root   sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

- The file type indicates what sort of file it is. It could be a regular file, a directory, or a special file like a device.

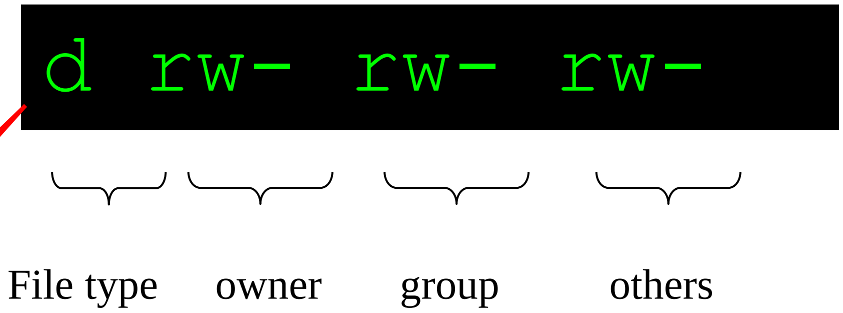


```
- rw- rw- rw-
```

```
$ ls -l
-rw-rw-rw- 1 root    sys        34 Jul 15 12:50 file1.txt
$
```

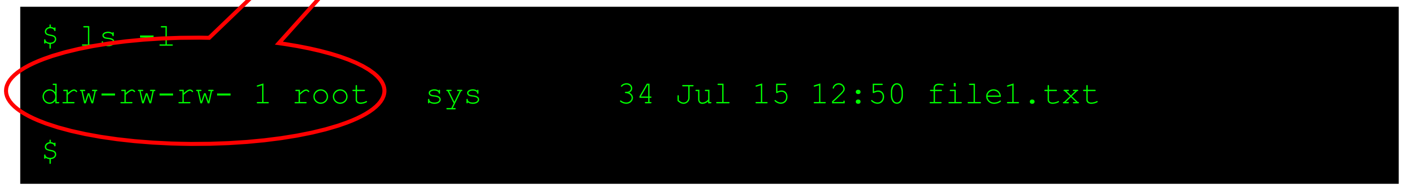
Permissions/File Access Modes

- ❑ A file type of “d” would indicate that this file is a directory.



```
d rw- rw- rw-
```

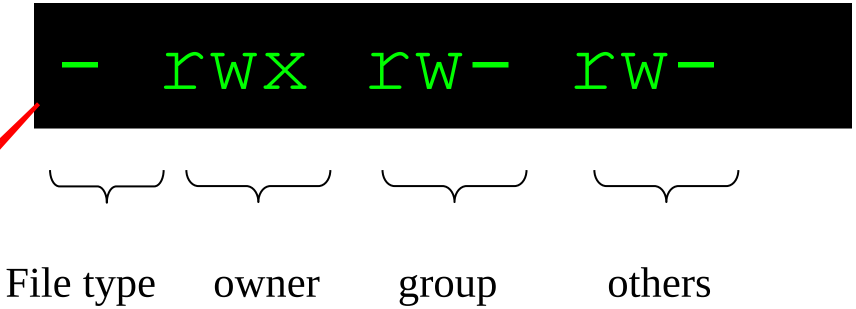
File type owner group others



```
$ ls -l
drw-rw-rw- 1 root   sys        34 Jul 15 12:50 file1.txt
$
```

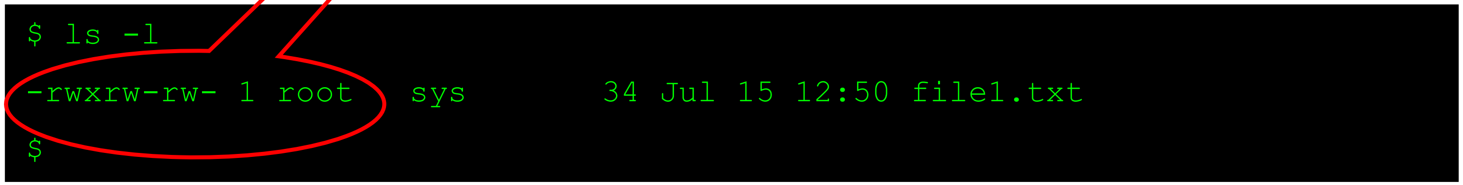
Permissions/File Access Modes

- The other permissions are grouped in three’s, one for read, one for write and one for execute. The absence of one of these means there is no permission.



```
- rwx rw- rw-
```

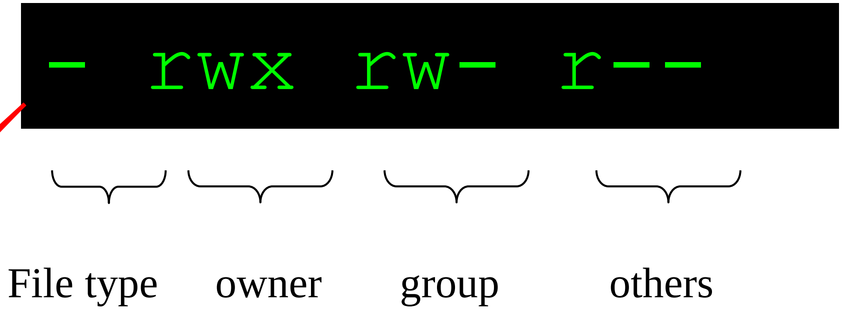
File type owner group others



```
$ ls -l
-rwxrw-rw- 1 root    sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

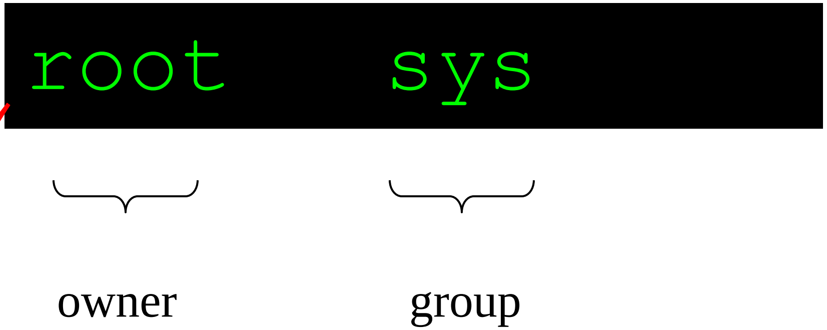
- ❑ This file has read, write and execute for the owner, read and write for anyone in the same group, and read only for anyone else (others).



```
$ ls -l
-rwxrw-rw- 1 root   sys        34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

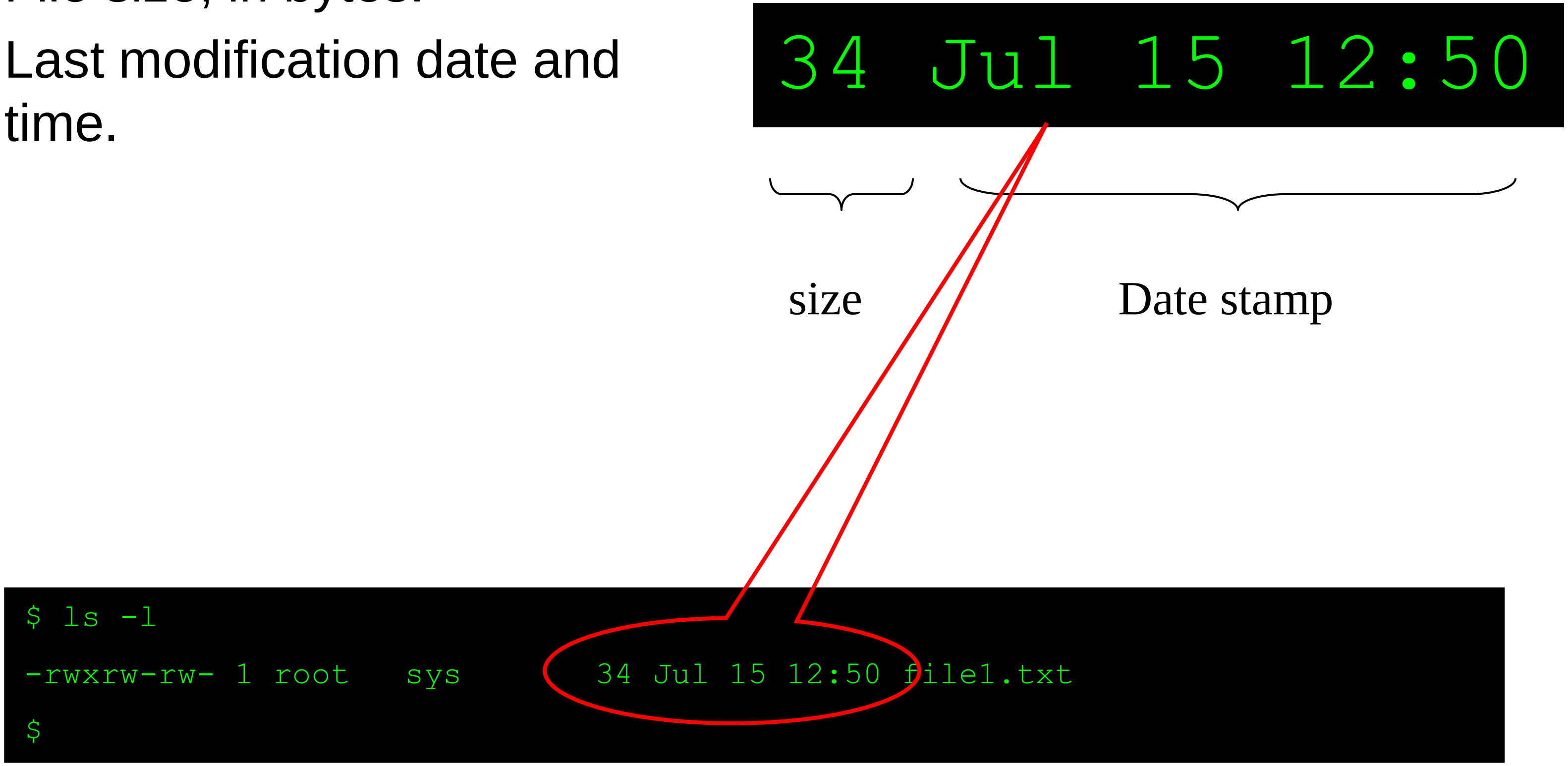
- ❑ These show who owns the file and to which group it belongs.
- ❑ Any user in this group has the group permissions seen earlier.



```
$ ls -l
-rwxrw-rw- 1 root   sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

- ❑ File size, in bytes.
- ❑ Last modification date and time.

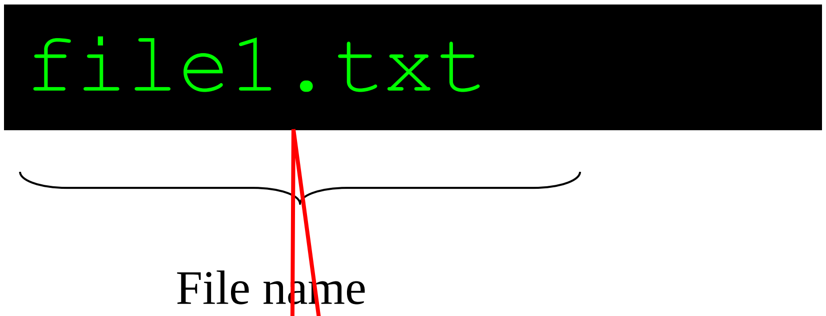


```
34 Jul 15 12:50
```

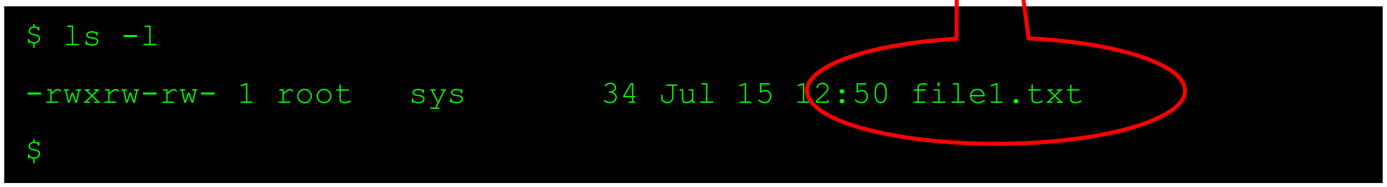
```
$ ls -l
-rwxrw-rw- 1 root   sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

- ❑ The name of the file itself, or filename.



```
file1.txt
```

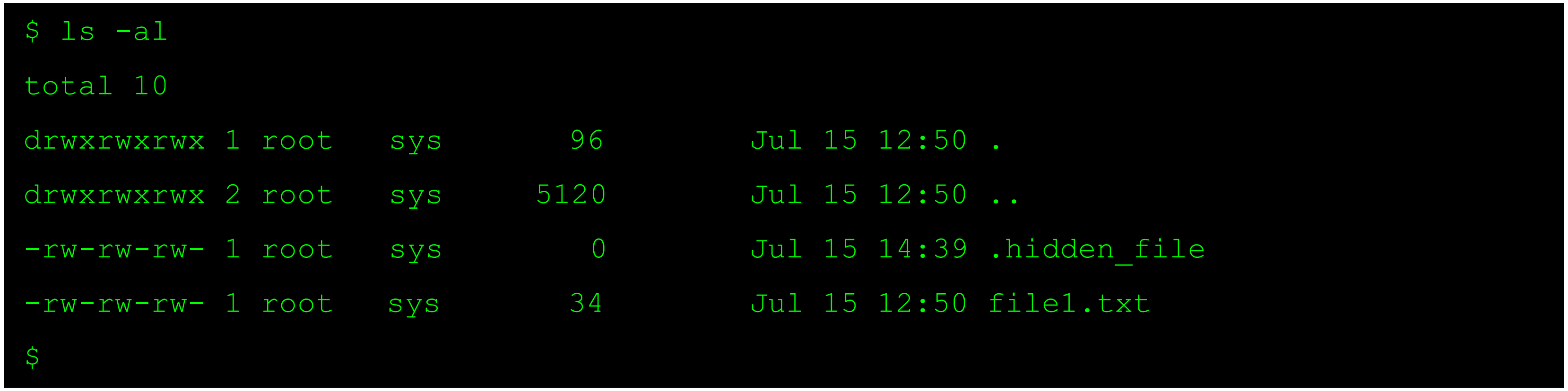


```
$ ls -l
-rwxrw-rw- 1 root   sys       34 Jul 15 12:50 file1.txt
$
```

Permissions/File Access Modes

❑Other options used with the ls command give more detail.

❑To see hidden files and a long listing use “ls -al”.



```
$ ls -al
total 10
drwxrwxrwx 1 root   sys       96       Jul 15 12:50 .
drwxrwxrwx 2 root   sys     5120       Jul 15 12:50 ..
-rw-rw-rw- 1 root   sys        0       Jul 15 14:39 .hidden_file
-rw-rw-rw- 1 root   sys       34       Jul 15 12:50 file1.txt
$
```

UNIX Fundamentals Part I

❑ UNIX History

❑ The Many Flavours' of UNIX

❑ The Structure of UNIX

❑ Access to UNIX Systems

❑ Processes

❑ **Filesystems & Directories**

- **Directory Structures**
- **Permissions/File Access Modes**
- **Directory & File Commands**

❑ Devices

Directory & File Commands

- ls command
- pwd command
- cd command
- cat, pg & more
- head & tail
- mv, cp & rm
- mkdir & rmdir
- chown, chgrp, chmod
- umask

Directory & File Commands

- ls - list contents of a directory
- pwd - print working (current directory)
- cd - change directory

```
$ ls -al
total 10
drwxrwxrwx 1 root   sys       96        Jul 15 12:50 .
drwxrwxrwx 2 root   sys     5120        Jul 15 12:50 ..
-rw-rw-rw- 1 root   sys        0        Jul 15 14:39 .hidden_file
-rw-rw-rw- 1 root   sys       34        Jul 15 12:50 file1.txt
$ pwd
/tmp/ld
$ cd ..
$ pwd
/tmp
```

Directory & File Commands

- We can use wildcards to list files with certain names.
  - *   Any number of any characters
  - ?   Any single character

```
$ ls -l file*
-rw-rw-rw- 1 root   sys       34        Jul 15 12:50 file1.txt
```

```
$ ls *.txt
file1.txt
```

```
$ ls -l f?le*
-rw-rw-rw- 1 root   sys       34        Jul 15 12:50 file1.txt
```

Directory & File Commands

- To see the contents of a file we use the “**cat**” command.

```
$ cat file1.txt
this is the contents of file1.txt
$
```

- For files that contain more data than will fit on a single screen we can use “**more**” or “**pg**” to show the data a screen at a time.

Directory & File Commands

❑To move a file use the "**mv**" command. This also serves as a renaming command.

❑To make a copy of a file, use "**cp**".

```
$ mv file1.txt File2.txt
$ mv File2.txt /tmp
$ cp /tmp/File2.txt .
$
```

Directory & File Commands

❑To delete a file, use the “rm” command.

❑**BEWARE :** *if you remove a file it is gone completely, there is no “recycle bin”!*

```
$ rm /tmp/File2.txt
$ ls /tmp/File*
$
```

Directory & File Commands

□Use “mkdir” to create a new directory.

□Use “rmdir” to remove a directory.

```
$ mkdir test
$ ls -ld test
drw-rw-rw- 1 root   sys       2046      Jul 15 15:02 test
$ rmdir test
$ ls -ld test
$
```

Directory & File Commands

- To change the owner of a file (providing you are the owner or “root”), use “**chown**”.

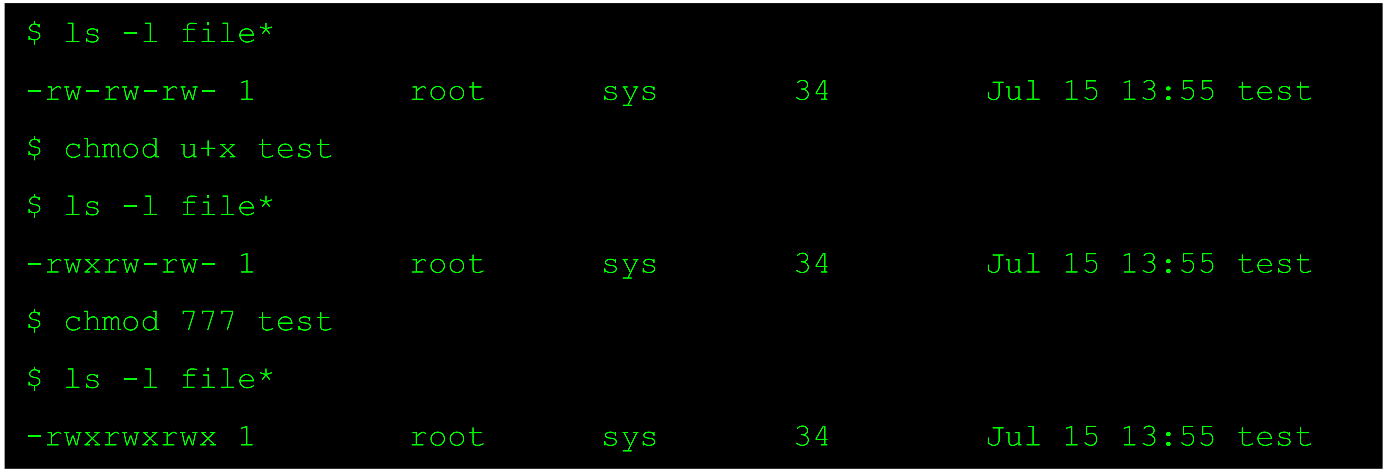
- To change the group of a file (providing you are the owner, have group permissions or are “root”), use “**chgrp**”.

```
$ ls -l file*
-rw-rw-rw- 1         root      sys       34        Jul 15 13:55 test
$ chown train6 test
$ chgrp staff test
$ ls -l file*
-rw-rw-rw- 1         train6    staff     34        Jul 15 13:55 test
$
```

Directory & File Commands

- To change the permissions of a file use "**chmod**".

- A symbolic notation can be used to select permissions or an octal value.



```
$ ls -l file*
-rw-rw-rw- 1        root      sys       34        Jul 15 13:55 test
$ chmod u+x test
$ ls -l file*
-rwxrw-rw- 1        root      sys       34        Jul 15 13:55 test
$ chmod 777 test
$ ls -l file*
-rwxrwxrwx 1        root      sys       34        Jul 15 13:55 test
```

Directory & File Commands

- Files and directories have permissions set as they are created.
- The permissions on newly created files/directories are set by the users profile and/or the **umask** filemode command
- **umask** can also be used to display the current file mode creation mask.

```
gbsrual0048:root:/opt/nmon/bin> umask
027

gbsrual0048:root:/opt/nmon/bin> umask -S

u=rwx,g=rx,o=
gbsrual0048:root:/opt/nmon/bin> umask a=rx,ug+w
gbsrual0048:root:/opt/nmon/bin> umask -S
u=rwx,g=rwx,o=rx
```

Filesystems & Directories Checkpoint (1)

❑ Match the various options of the ls command with their functions:

- **-a**
- **-t**
- **-d**
- **-r**
- **-l**

[_]   - Provides a long listing of files

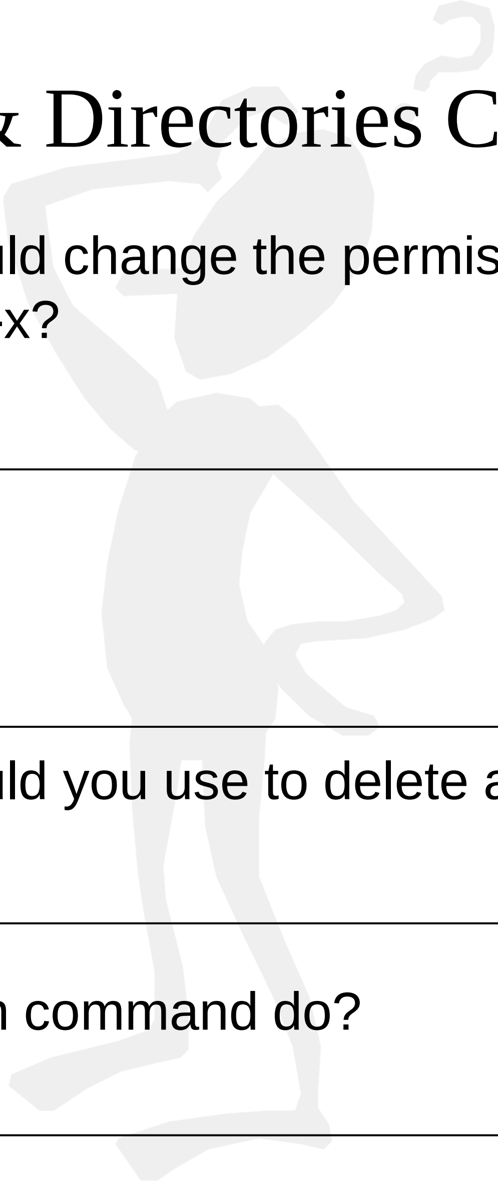
[_]   - Will list hidden files

[_]   - List subdirectories & their contents

[_]   - Sort the output by last date/time modification

[_]   - Displays information about a directory in reverse order

Filesystems & Directories Checkpoint (2)



- □ What command would change the permissions of a file from rwxrwxr-x to rwxr-xr-x?

____

- □ What is the umask?

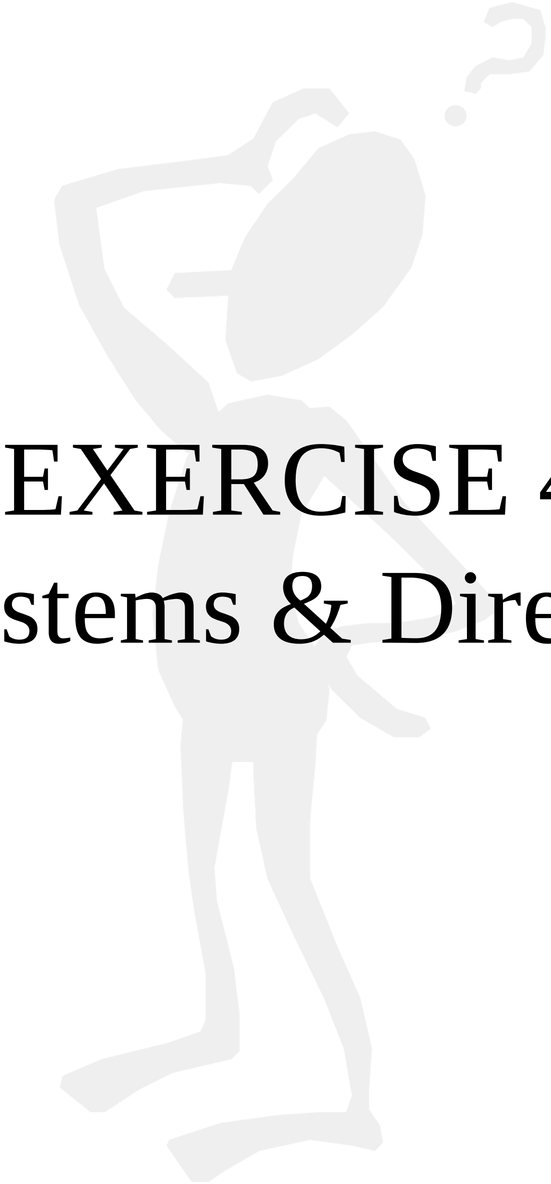
____

- □ What command would you use to delete a directory?

____

- □ What does the touch command do?

____



EXERCISE 4
Filesystems & Directories

UNIX Fundamentals Part I

- □ UNIX History
- □ The Many Flavours' of UNIX
- □ The Structure of UNIX
- □ Access to UNIX Systems
- □ Processes
- □ Filesystems & Directories
- □ **Devices**

Devices I

- ❑ In UNIX everything is a file.
- ❑ Resources within a UNIX/UNIX platform are accessed through file names as opposed to drive letters.
- ❑ All hardware is represented as device files.
  - ▪ **Example: /dev/cd0 can be a cdrom.**
- ❑ All of the devices for the computers resources reside in its devices directory, usually /dev.
- ❑ Devices include
  - ▪ **Hard disk (SCSI & IDE)**
  - ▪ **Tape drives**
  - ▪ **CDROMs**
  - ▪ **Floppy disks**
  - ▪ **ISDN terminal adapters / modems**
  - ▪ **mice**
  - ▪ **terminals (i.e. the screen)**
  - ▪ **keyboard**

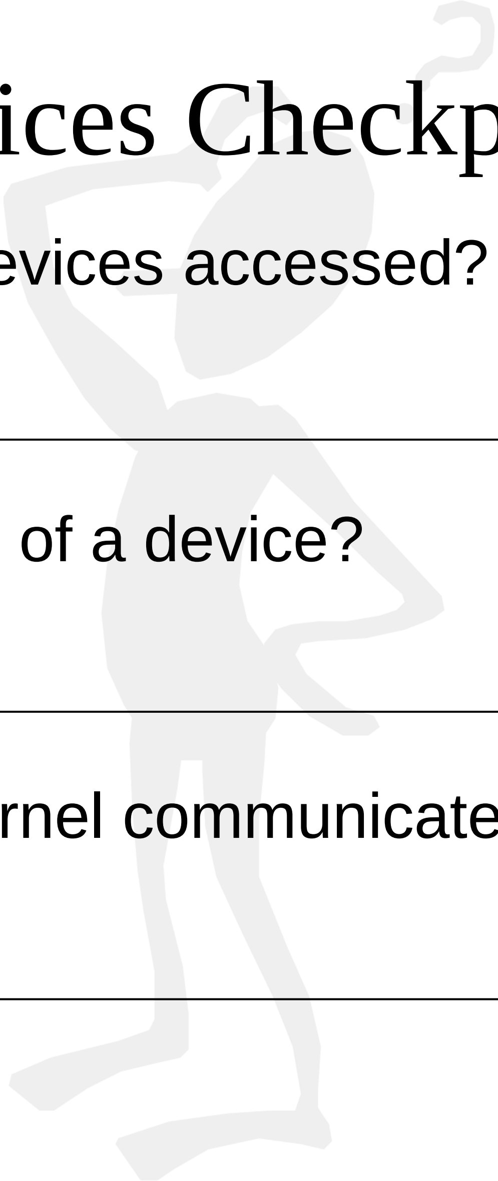
Devices II

- ❑ A device driver is a program that manages the interaction between a piece of hardware and the kernel.
- ❑ Device drivers implement a standardized set of function with a device interacts with the kernel
- ❑ Functions include
  - **Open, close, read, stop, write, timeout**
- ❑ Devices fall into two principle types
  - **Block devices read and write block (usually multiples of 512 bytes)**
  - **Character devices can read and write one byte at a time Different UNIX platforms have different naming conventions for these devices.**
- ❑ The kernel maintains tables for the block and character devices
- ❑ When programs perform operations on devices the kernel directs the control to the correct function in a device driver

Device Examples

	AIX	*HPUX10*	*Solaris*	*DEC OSF/1*	*IRIX*	*Linux*	*SCO*
Floppy	/dev/rfd0	/dev/rfloppy/c#t#d0	/dev/diskette	/dev/rfd0	/dev/rdsk/fds0d2.3.5	/dev/fd0	/dev/rfd0
CDROM	/dev/cd0	/dev/dsk/c#t#d0	/dev/dsk/c0t#d0s0	/dev/rz#c	/dev/scsi/sc0d#l0	/dev/cdrom	/dev/cd0
Hard disk	/dev/hdisk1	/dev/dsk/c0t2d0	/dev/dsk/c0t0d0s2	/dev/rz2c	/dev/dsk/dks0d2s7	/dev/sda	/dev/hd01

Devices Checkpoint



❑ How are UNIX devices accessed?

____

❑ Give an example of a device?

____

❑ How does the kernel communicate to devices?

____

Fin

(End of Part I)

Thank you for attending!

Any Questions?