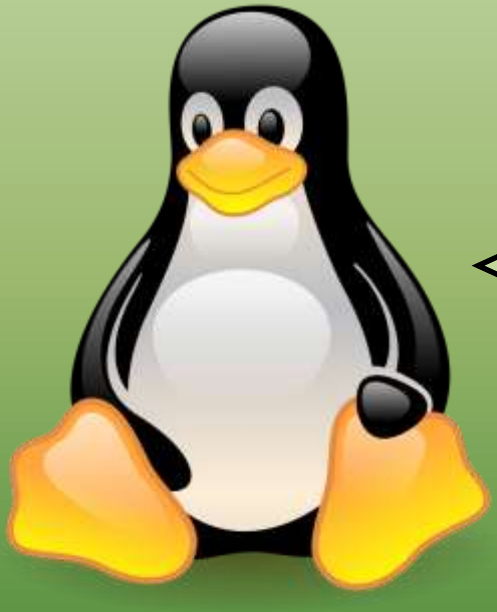
Linux in Detail

Damian Gordon

Linux in Detail



This is "Tux", the Linux mascot. It was originally created as an entry to a Linux logo competition. Tux is the most commonly used icon for Linux.

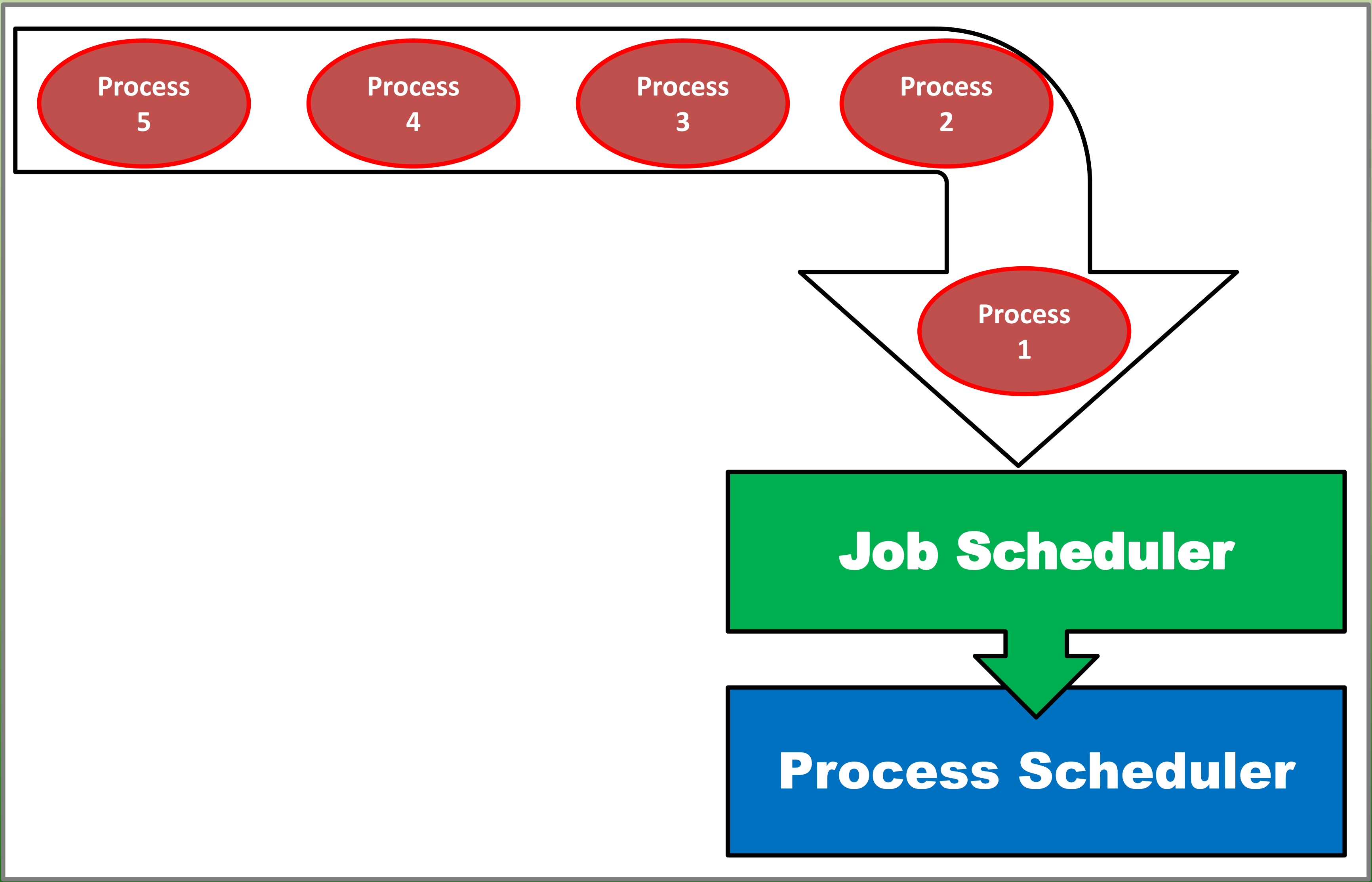
Damian Gordon

Linux in Detail

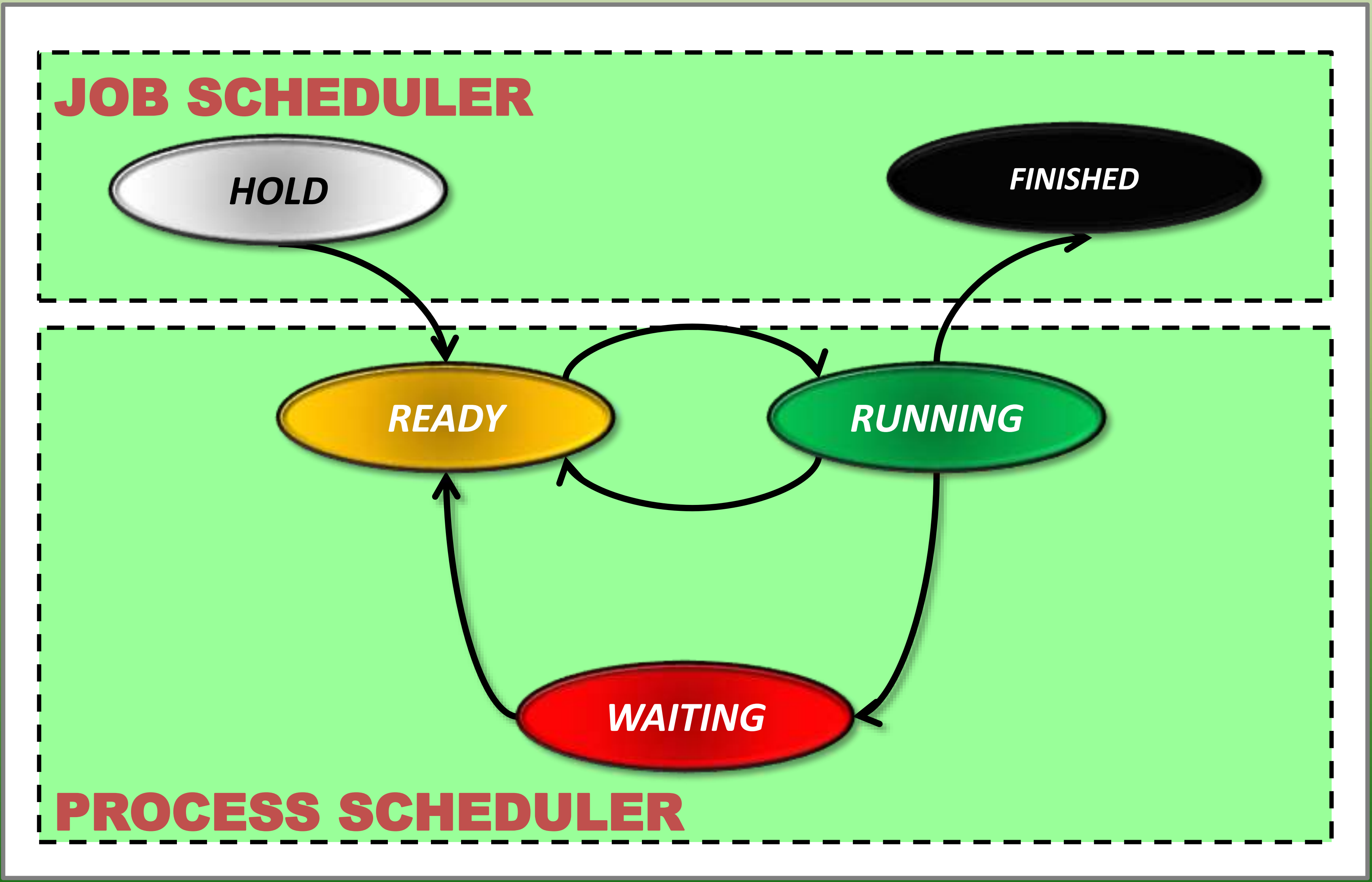
- Let’s look at:
  - Processor Management
  - File Management
  - Memory Management
  - Device Management
  - Command Line Interface

Processor Management

Linux: Processor Management



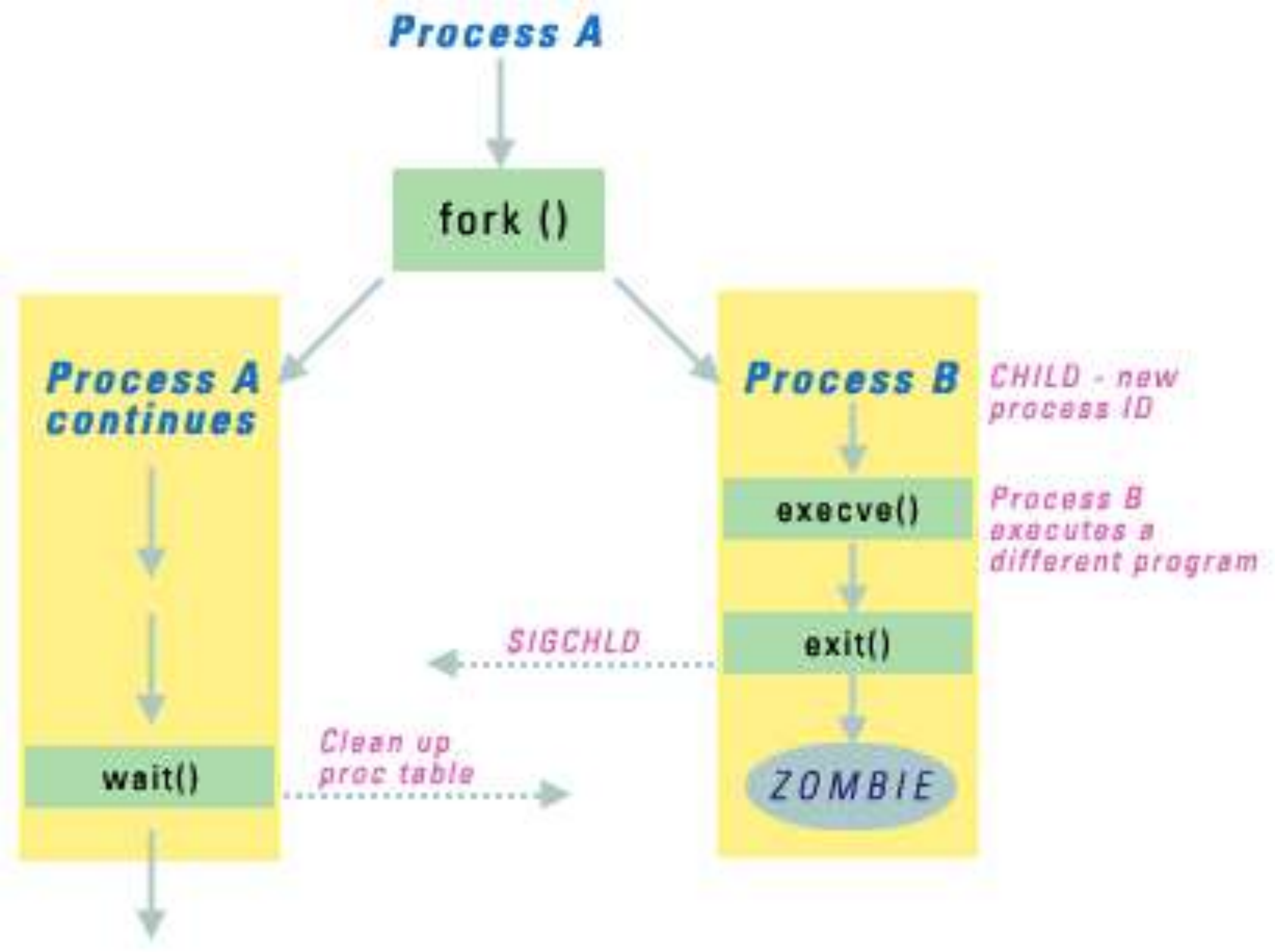
Linux: Processor Management



Linux: Processor Management

- **fork()**
- Linux uses the same parent-child process management found in Unix, centring on the `fork()` command.
- `fork()` gives the user to create a copy of an executing program.
- This command gives the second program all the attributes of the first program, such as any open files, and saves the first program in its original form.

Linux: Processor Management



Linux: Processor Management

- The system call `fork()` splits a program into two copies, which are both running from the statement after the fork command.
- The original process (Process A) is called the **parent process** and the resulting process (Process B) is the **child process**. A child inherits the parent’s open files.
- When `fork()` is executed, the **child process** gets a new process id (called *pid* for short), this is done in a way that ensures that each process has its own unique ID number.

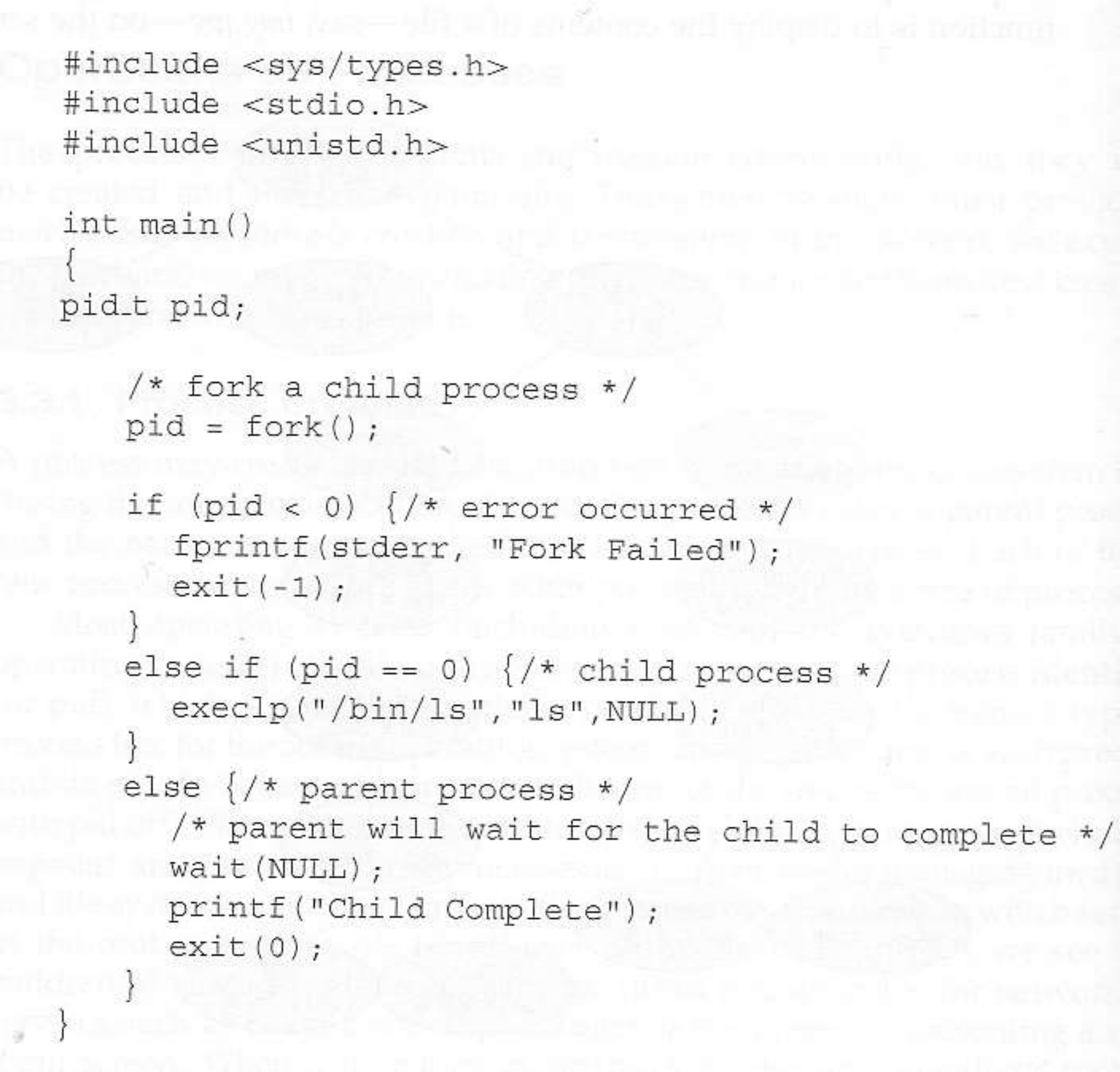
Linux: Processor Management

- **exec()**
- Alternatively, the exec family of commands—`execl()`, `execv()`, `execle()`, `execlp()`, and `execlvp()`—is used to start execution of a new program from another program, but unlike `fork()`, which results in two processes running the same program in memory, a successful `exec()` call will lay the second program over the first, leaving only the second program in memory.
- So `exec()` changes what the program is doing, but doesn't change the process id (pid).

Linux: Processor Management

- **So often you do a fork() followed by an exec() on the child process…**

Linux: Processor Management



```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

   /* fork a child process */
   pid = fork();

   if (pid < 0) {/* error occurred */
     fprintf(stderr, "Fork Failed");
     exit(-1);
   }
   else if (pid == 0) {/* child process */
     execlp("/bin/ls","ls",NULL);
   }
   else {/* parent process */
     /* parent will wait for the child to complete */
     wait(NULL);
     printf("Child Complete");
     exit(0);
   }
}
```

Linux: Processor Management

Linux: Processor Management

- The Linux process scheduler typically scans the list of processes in the READY state and, using predefined criteria, chooses which process to execute.
- The scheduler has three different scheduling types: two for real-time processes and one for normal processes.

Linux: Processor Management

Name	Priority Level	Process Type	Scheduling Policy
SCHED_FIFO	Highest Priority	For non-pre-emptable real-time processes.	First In, First Out (FIFO)
SCHED_RR	Medium Priority	For pre-emptable real-time processes.	Round Robin and priority
SCHED_OTHER	Lowest Priority	For normal processes.	Priority only

Linux: Processor Management

- **SCHED_FIFO**
- From among the processes with the highest priority, the scheduler selects the process with the highest priority and executes it using the first in, first out algorithm. This process is normally not pre-emptible and runs to completion.

Linux: Processor Management

- **SCHED_RR**
- When executing a process of the second type, the scheduler chooses from this group with the highest priority and uses a round robin algorithm with a small time quantum, and when the time expires, other processes (such as a FIFO or another RR type with a higher priority) may be selected and executed before the first process is allowed to run to completion.

Linux: Processor Management

- **SCHED_OTHER**
- The third type of process has the lowest priority and is executed only when there are no processes with higher priority in the READY queue.

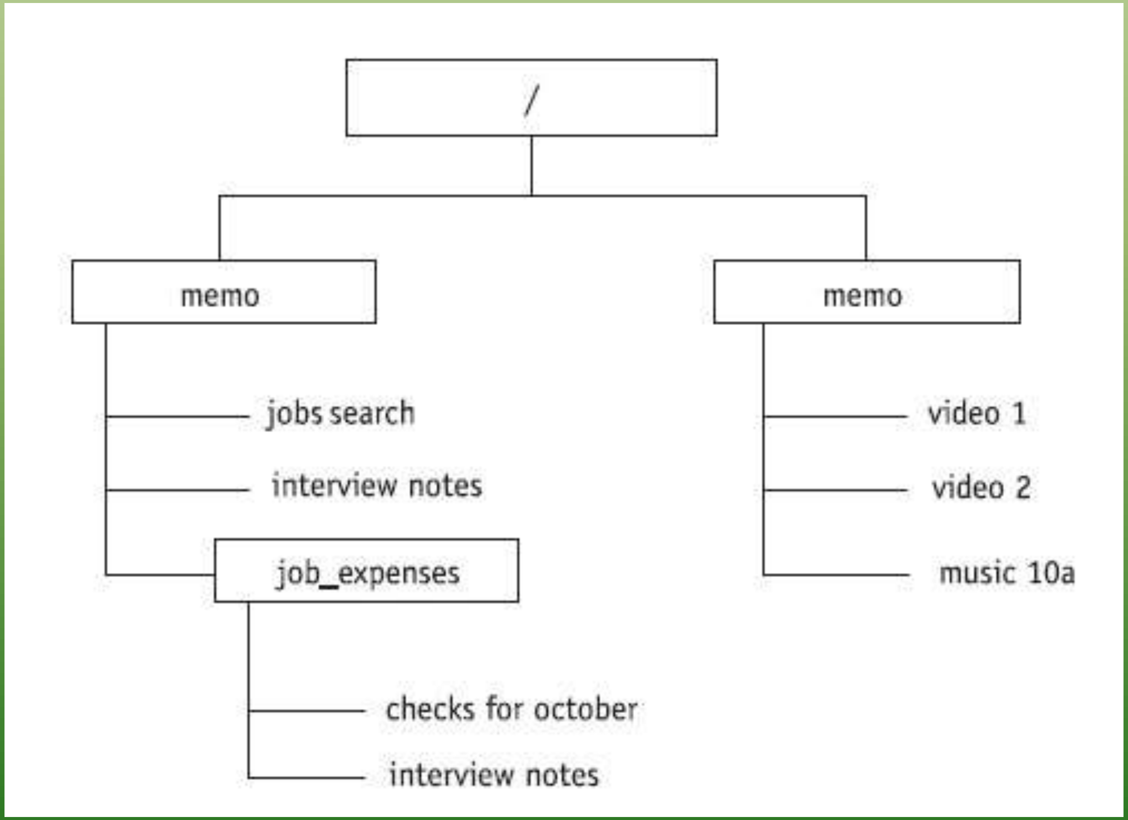
File Management

Linux: File Management

- Linux is case sensitive, so the following are different files:
  - README.TXT
  - ReadMe.TXT
  - readMe.TXT
  - readme.TXT

Linux: File Management

- A typical Linux file structure is:

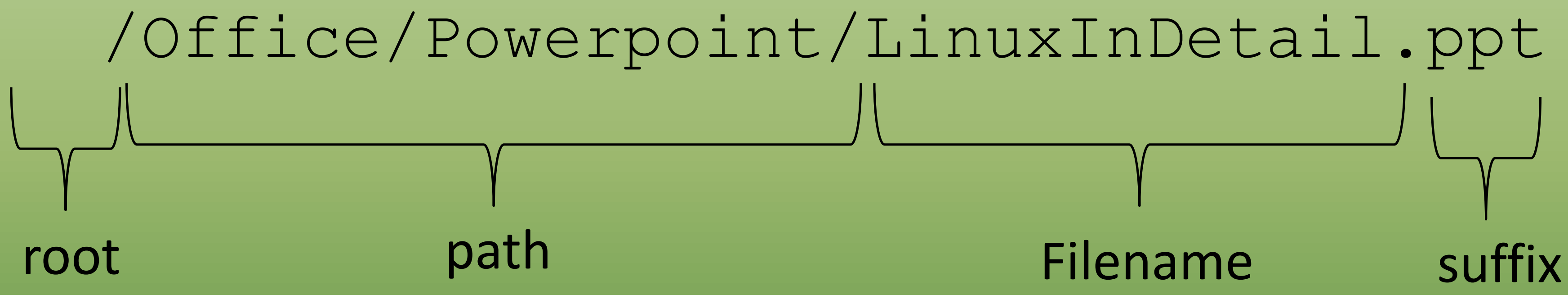


Linux: File Management

- Filenames can be up to 255 characters long and can contain alphabetic characters, underscores, and numbers.
- File suffixes (which is the Linux term for file extensions) are optional.
- Filenames can include a space; however, this can cause complications if you're running programs from the command line because a program named *interview notes* would be viewed as a command to run two files: *interview* and *notes*. To avoid confusion, the two words can be enclosed in quotes: *"interview notes."*

Linux: File Management

- The full filename includes path information:



```
/Office/Powerpoint/LinuxInDetail.ppt
```

root

path

Filename

suffix

Linux: File Management

- In Linux the Access Controls are:
    - R: Read
    - W: Write
    - X: Execute

Linux: File Management

- In Linux access to a file can assigned to one of three groups:

- User
- User Group
- World

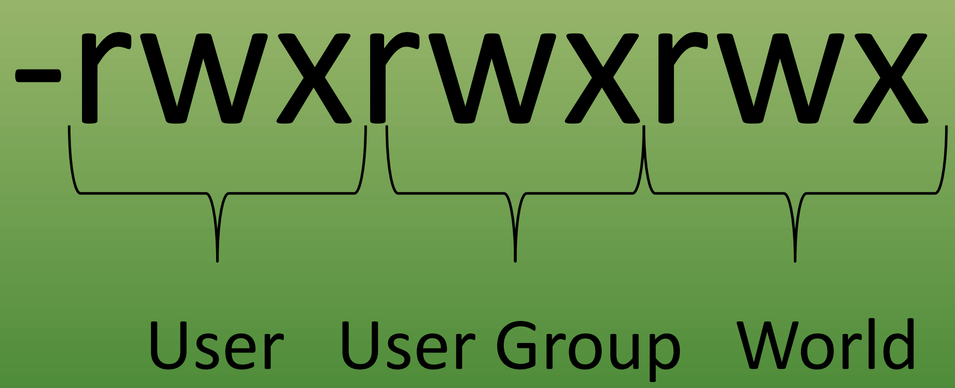
Linux: File Management

- In Linux access to a file can assigned to one of three groups:

- User -you
- User Group – everyone in your group
- World – everyone with a login to the system

Linux: File Management

- In Linux access to a file can assigned to one of three groups:



- -rwxrwxrwx
- User User Group World

Linux: File Management

- In Linux access to a file can assigned to one of three groups:
- -rwxrwxrwx
- -111111111

Linux: File Management

- In Linux access to a file can assigned to one of three groups:
- -rwxr-xr-x
- -111101101

Linux: File Management

- In Linux access to a file can assigned to one of three groups:
- -rwx--x--x
- -101001001

Linux: File Management

- In Linux access to a file can assigned to one of three groups:
- -rwxrwxrwx
- -111111111
- - 7 7 7

Linux: File Management

- In Linux access to a file can assigned to one of three groups:
- -rwxr-xr-x
- -111101101
- - 7 5 5

Linux: File Management

- In Linux access to a file can assigned to one of three groups:
- -rwx--x--x
- -111001001
- - 7   1   1

Linux: File Management

- If we want to grant permissions to file, e.g. MakeABackup.bat, we do:

- **`chmod 755 MakeABackup.sh`**
- **`chmod 777 MakeABackup.sh`**
- **`chmod 700 MakeABackup.sh`**

Linux: File Management

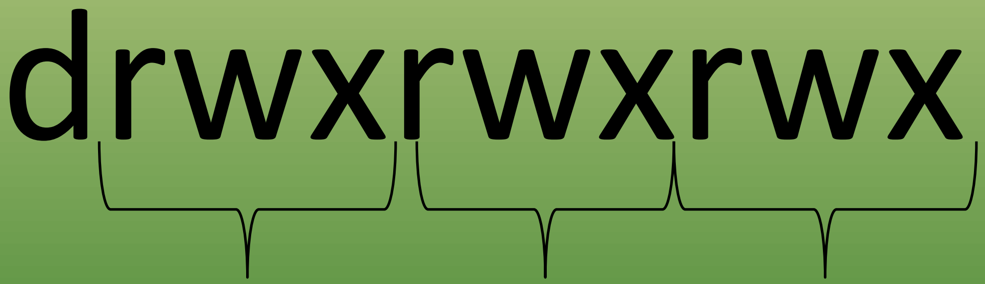
- In Linux access to a file can assigned to one of three groups:

$$-\underbrace{rwx}\underbrace{rwx}\underbrace{rwx}$$

- User User Group World

Linux: File Management

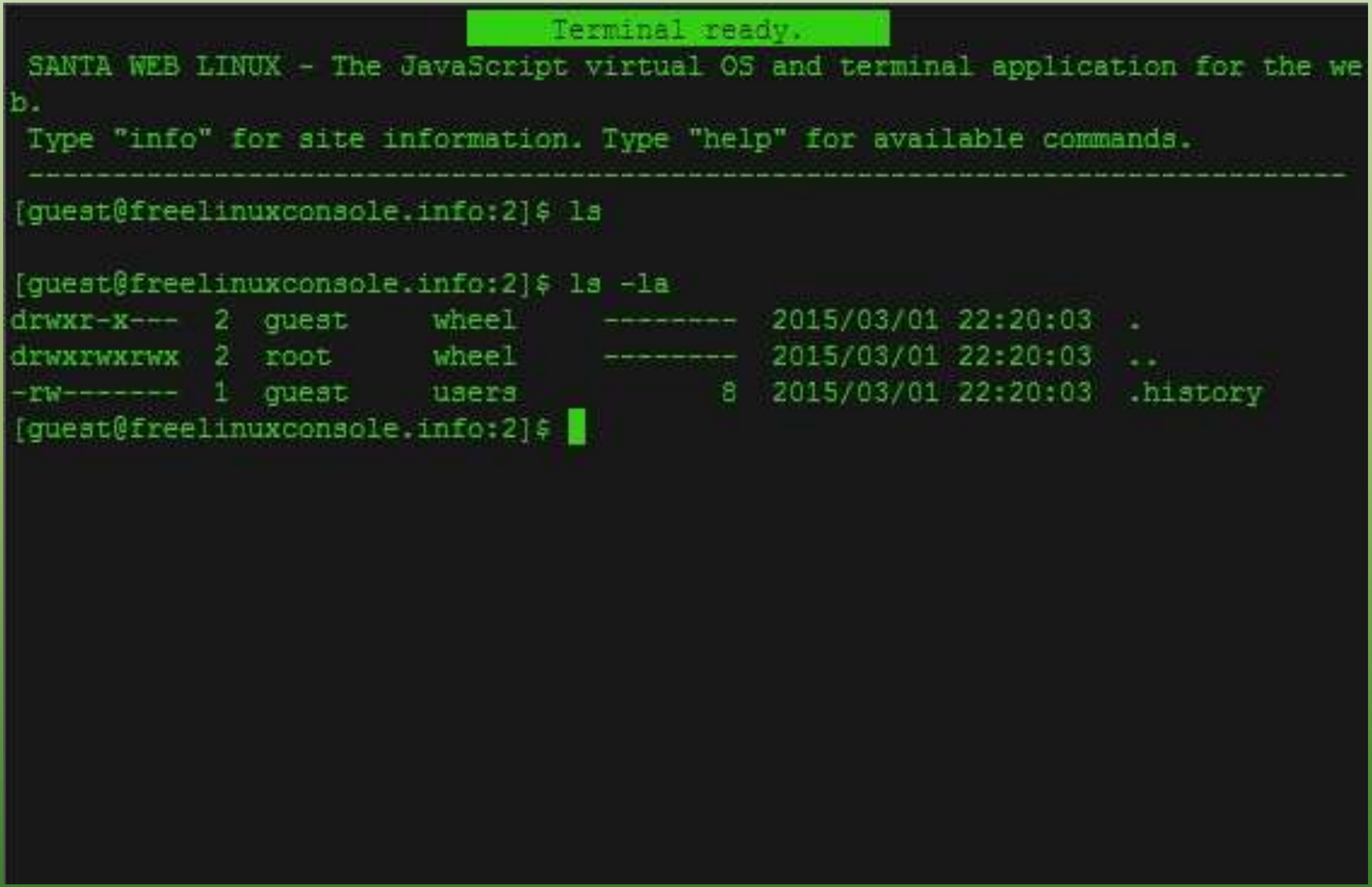
- In Linux access to a file can assigned to one of three groups:



drwxrwxrwx

- User User Group World

Access Control Matrix



```
                          Terminal ready.
 SANTA WEB LINUX - The JavaScript virtual OS and terminal application for the we
b.
 Type "info" for site information. Type "help" for available commands.
 ------------------------------------------------------------------------------
[guest@freelinuxconsole.info:2]$ ls

[guest@freelinuxconsole.info:2]$ ls -la
drwxr-x---  2  guest     wheel     --------  2015/03/01 22:20:03  .
drwxrwxrwx  2  root      wheel     --------  2015/03/01 22:20:03  ..
-rw-------  1  guest     users            8  2015/03/01 22:20:03  .history
[guest@freelinuxconsole.info:2]$ 
```

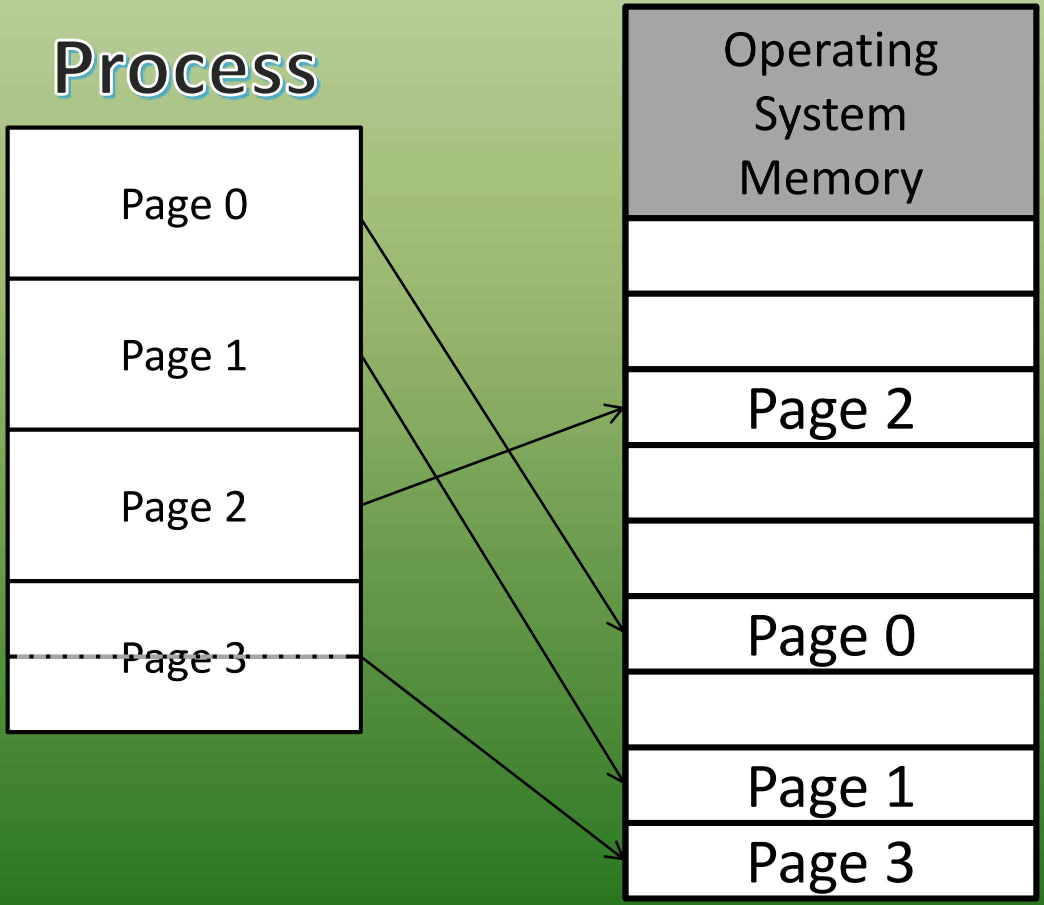
Memory Management

Linux: Memory Management

- Linux allocated 1GB for the kernel, and 3GB for executing processes.
- The 3GB address space is divided into:
  - Process code
  - Process data
  - Shared library data used by processes
  - Stack used by process

Linux: Memory Management

- When thinking about virtual memory we'll remember that the operating system divides a process into pages, and it divides main memory into page frames.



Linux: Memory Management

- When a process requests pages, Linux loads them into memory.
- When the kernel needs memory space, the pages are released on a Least-Recently Used (LRU) basis.
- To keep track of free and busy pages, Linux uses a system of page tables.

Linux: Memory Management

- Each virtual address in memory is stored as four elements:
  - Main Directory
  - Middle Directory
  - Page Table Directory
  - Page Frame

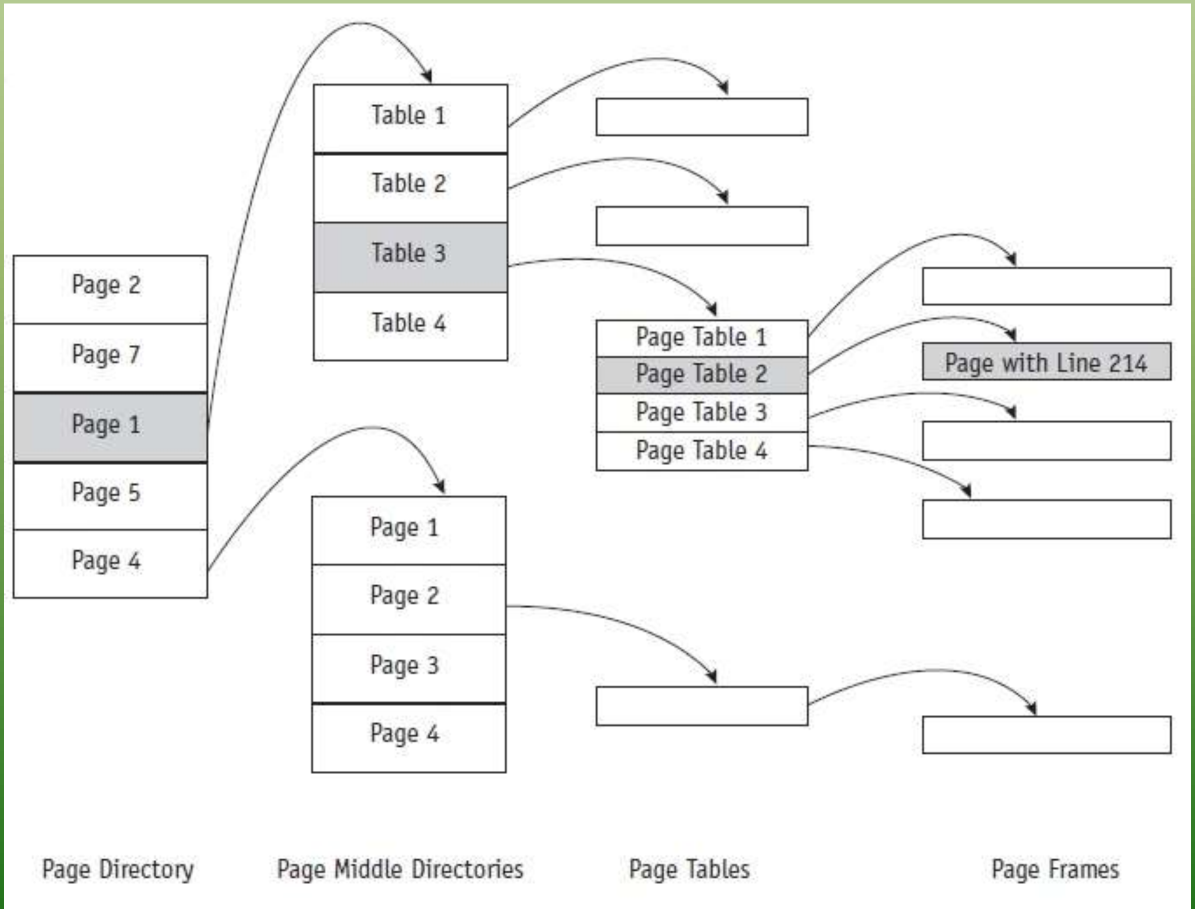
Linux: Memory Management

- Each virtual address in memory is stored as four elements:
  - – Main Directory
  - – Middle Directory
  - – Page Table Directory
  - – Page Frame

Example
Page 1
Table 3
Page Table 2
Location of Line 214

Linux: Memory Management

- As a diagram:



Device Management

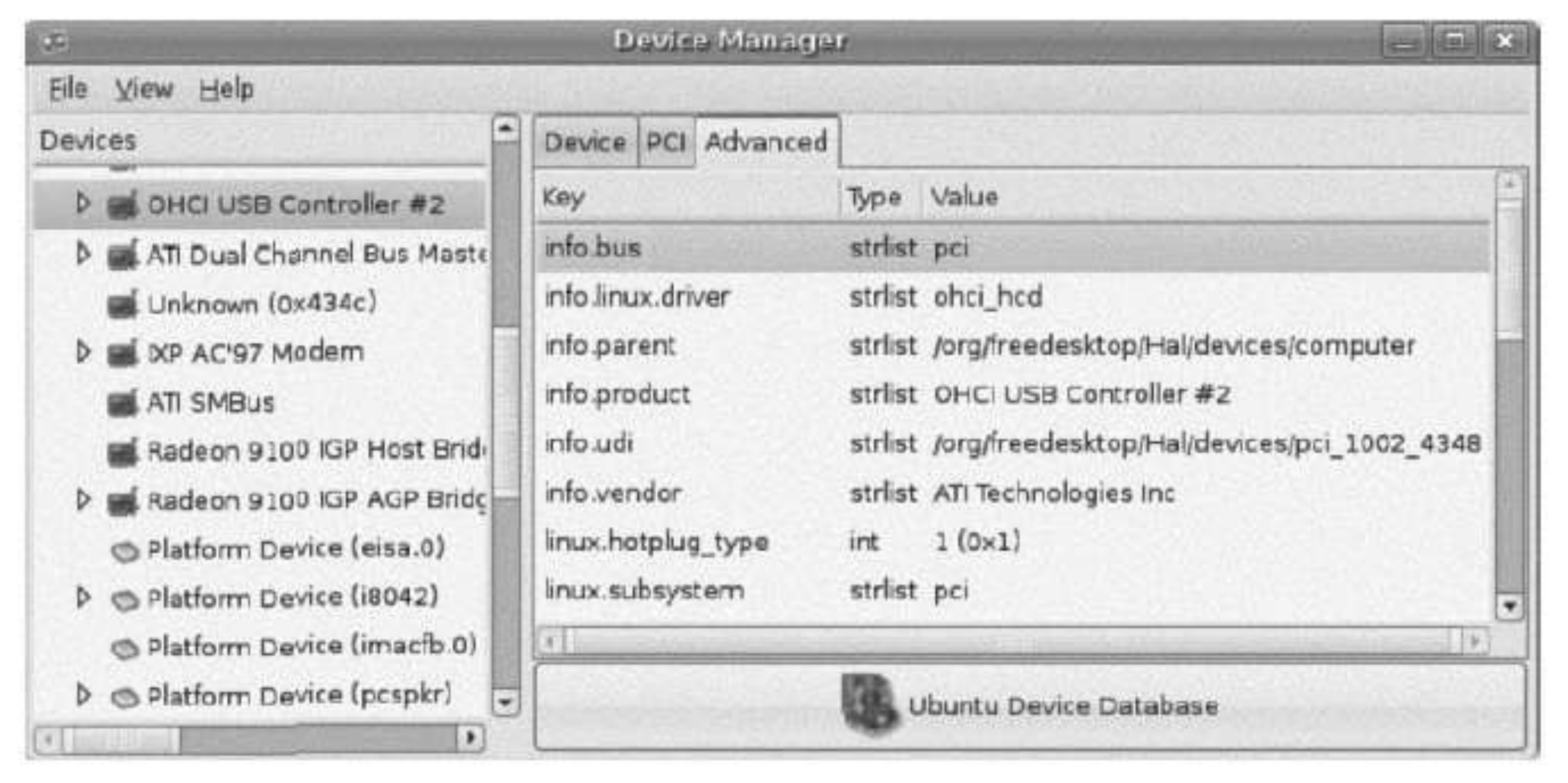
Linux: Device Management

- Linux is **device independent**, which improves its portability from one system to another.
- **Device drivers** supervise the transmission of data between main memory and the peripheral unit.

Linux: Device Management

- Linux treats devices as if they are files, and you can access devices the same way you access files in Linux.
- Devices are assigned not only a name but also descriptors that further identify each device and are stored in the device directory.

Linux: Device Management



- OHCI USB Controller #2
- ATI Dual Channel Bus Maste
- Unknown (0x434c)
- IXP AC'97 Modem
- ATI SMBus
- Radeon 9100 IGP Host Brid
- Radeon 9100 IGP AGP Bridg
- Platform Device (eisa.0)
- Platform Device (i8042)
- Platform Device (imacfb.0)
- Platform Device (pcspkr)

Key	Type	Value
info.bus	strlist	pci
info.linux.driver	strlist	ohci_hcd
info.parent	strlist	/org/freedesktop/Hal/devices/computer
info.product	strlist	OHCI USB Controller #2
info.udi	strlist	/org/freedesktop/Hal/devices/pci_1002_4348
info.vendor	strlist	ATI Technologies Inc
linux.hotplug_type	int	1 (0x1)
linux.subsystem	strlist	pci

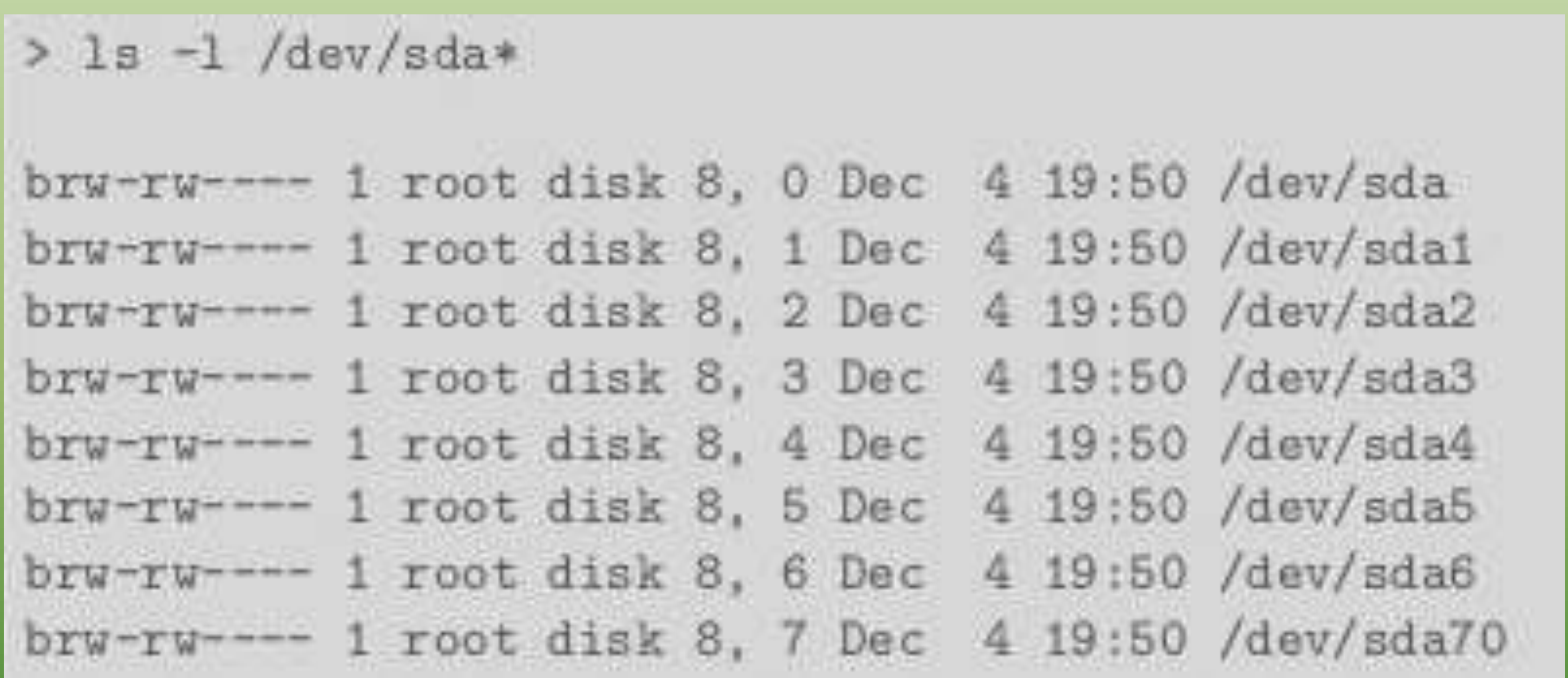
Linux: Device Management

- A **device driver** (or **driver**) is a computer program that operates or controls a particular type of device that is attached to a computer.
- A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used.

Linux: Device Management

- Linux identifies each device by a major device number and a minor device number.
  - the *major device number* identifies the driver associated with the device.
  - the *minor device number* is used by the kernel to determine exactly which device is being referred to.

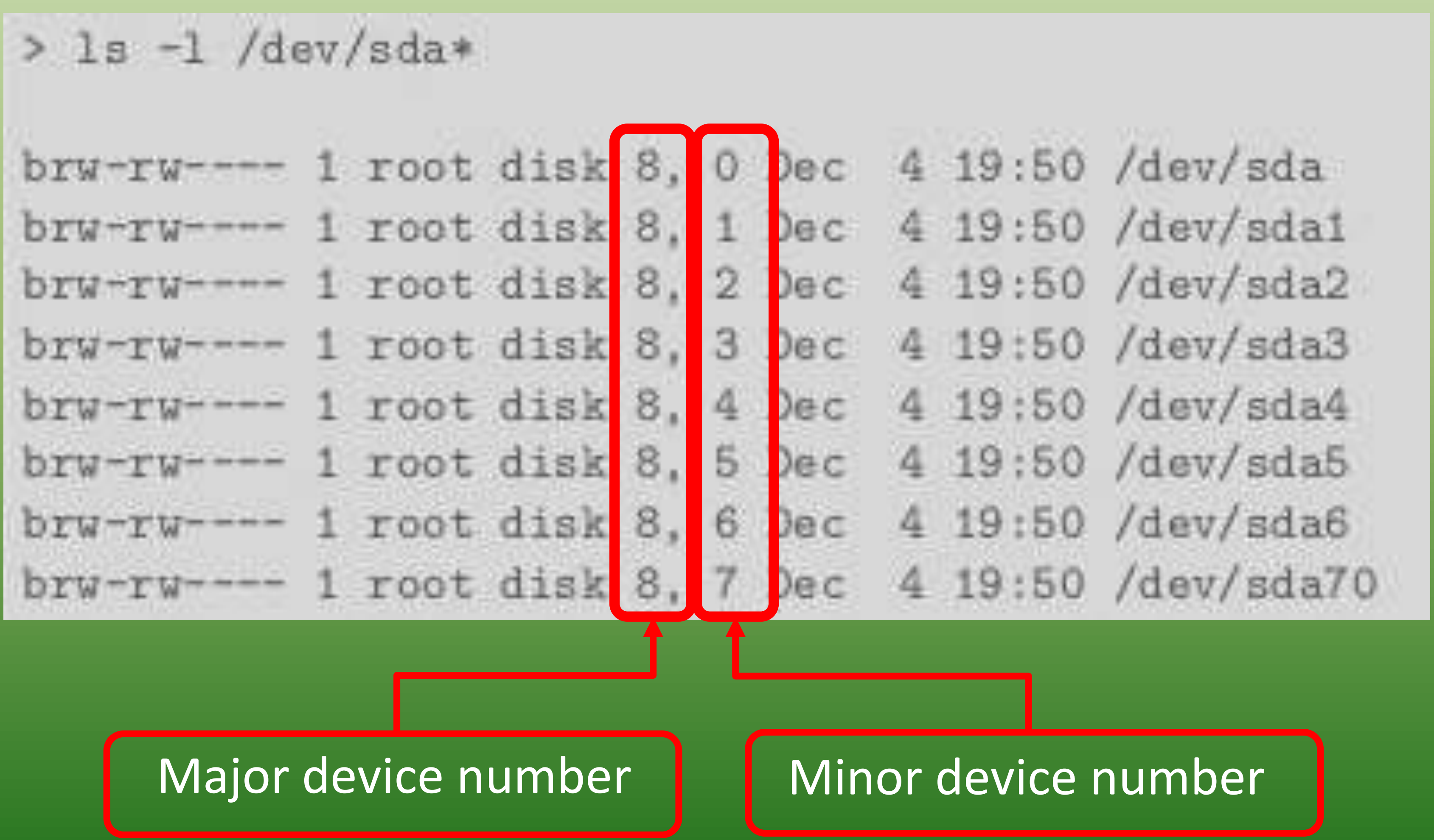
Linux: Device Management



```
> ls -l /dev/sda*

brw-rw---- 1 root disk 8, 0 Dec  4 19:50 /dev/sda
brw-rw---- 1 root disk 8, 1 Dec  4 19:50 /dev/sda1
brw-rw---- 1 root disk 8, 2 Dec  4 19:50 /dev/sda2
brw-rw---- 1 root disk 8, 3 Dec  4 19:50 /dev/sda3
brw-rw---- 1 root disk 8, 4 Dec  4 19:50 /dev/sda4
brw-rw---- 1 root disk 8, 5 Dec  4 19:50 /dev/sda5
brw-rw---- 1 root disk 8, 6 Dec  4 19:50 /dev/sda6
brw-rw---- 1 root disk 8, 7 Dec  4 19:50 /dev/sda70
```

Linux: Device Management



```
> ls -l /dev/sda*

brw-rw---- 1 root disk 8, 0 Dec  4 19:50 /dev/sda
brw-rw---- 1 root disk 8, 1 Dec  4 19:50 /dev/sda1
brw-rw---- 1 root disk 8, 2 Dec  4 19:50 /dev/sda2
brw-rw---- 1 root disk 8, 3 Dec  4 19:50 /dev/sda3
brw-rw---- 1 root disk 8, 4 Dec  4 19:50 /dev/sda4
brw-rw---- 1 root disk 8, 5 Dec  4 19:50 /dev/sda5
brw-rw---- 1 root disk 8, 6 Dec  4 19:50 /dev/sda6
brw-rw---- 1 root disk 8, 7 Dec  4 19:50 /dev/sda70
```

Major device number

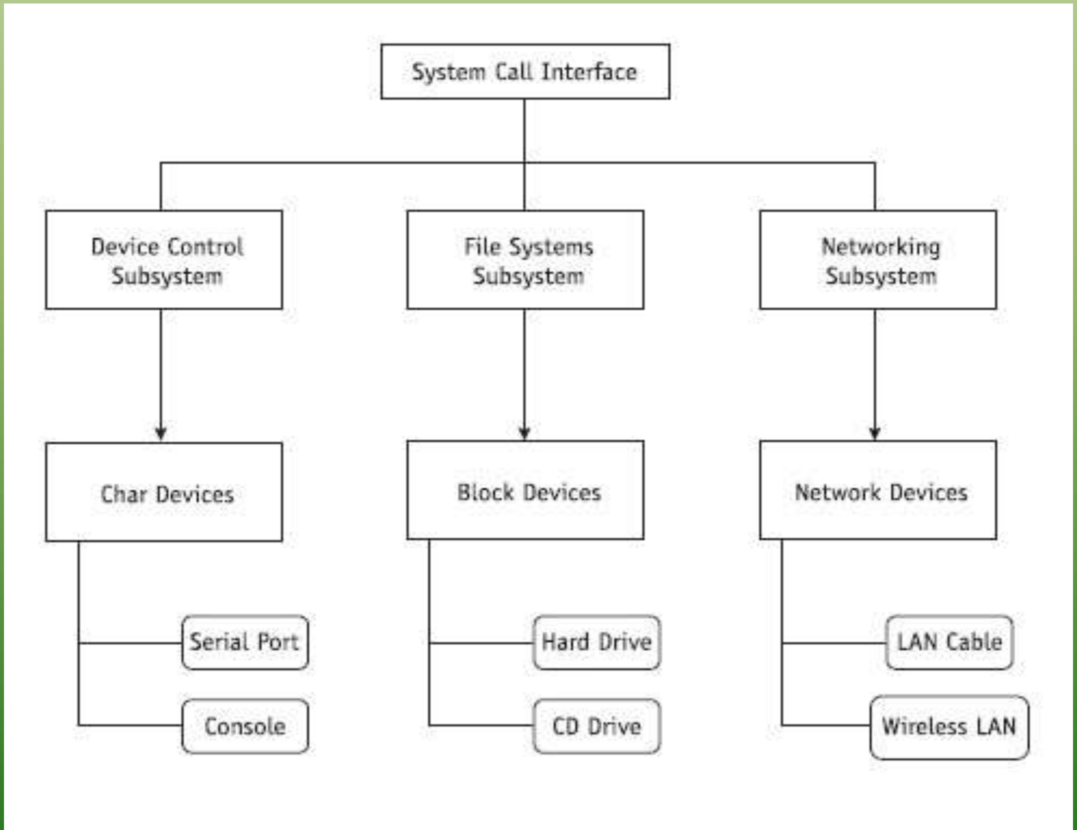
Minor device number

Linux: Device Management

- Standard versions of Linux often provide a comprehensive collection of common device drivers; but if the computer system should include hardware or peripherals that are not on the standard list, their device drivers can be retrieved from another source and installed separately.
- Alternatively, a computer programmer can write a device driver and install it for use.

Linux: Device Management

- Classes of device drivers:



Linux: Device Management

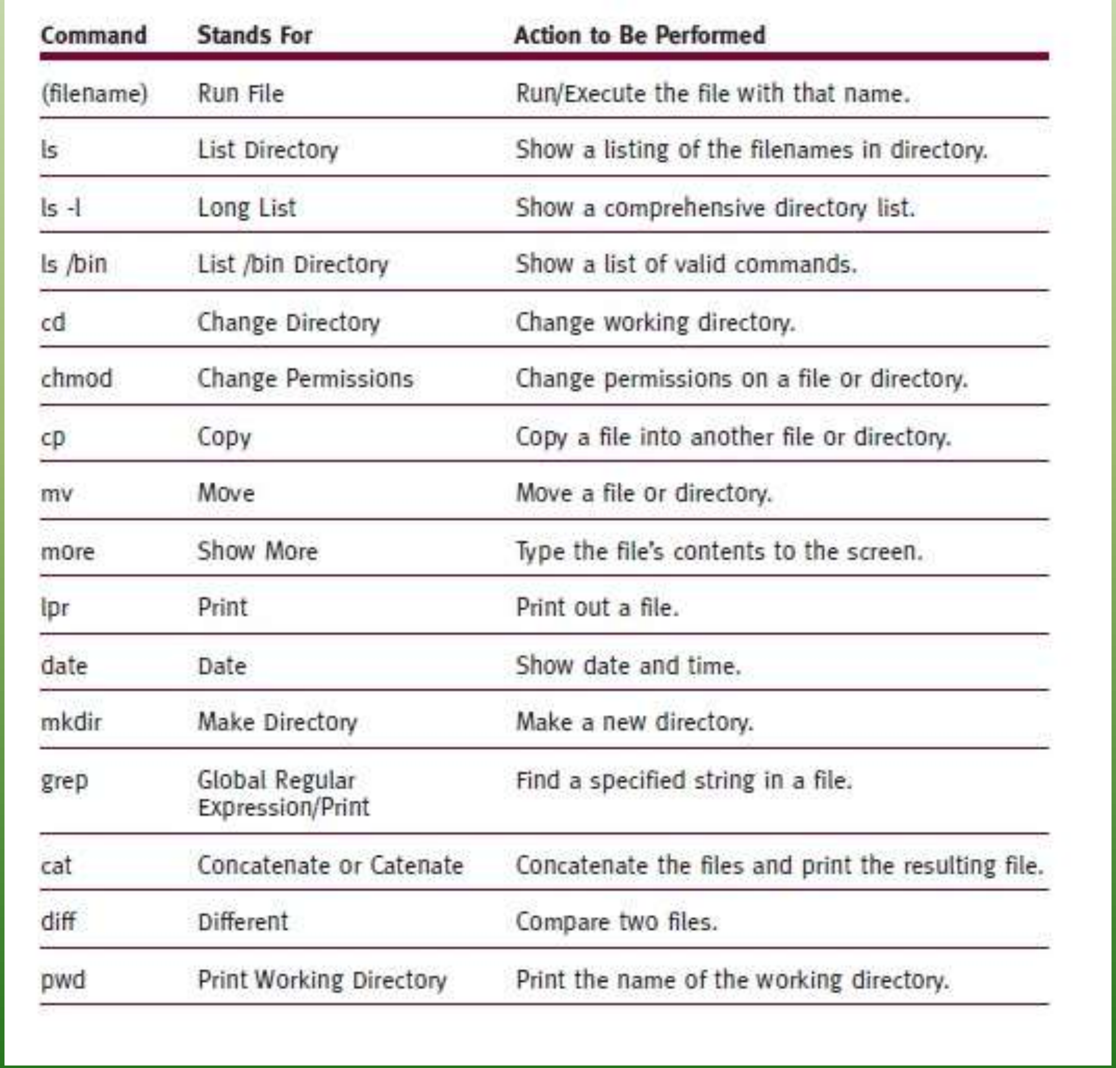
- **Char Devices**: Character devices are those that can be accessed as a stream of bytes, such as a communications port, monitor, or other byte-stream-fed device.
- **Block Devices**: Similar to char devices except that they can host a file system, such as a hard disk.
- **Network Devices**: Their function is to send and receive packets of information as directed by the network subsystem of the kernel.

Linux: Device Management

- A notable feature of Linux is its ability to accept new device drivers on the fly, while the system is up and running.
- That means administrators can give the kernel additional functionality by loading and testing new drivers without having to reboot each time to reconfigure the kernel.

Command Line Interface

Linux: Command Line Interface



Command	Stands For	Action to Be Performed
(filename)	Run File	Run/Execute the file with that name.
ls	List Directory	Show a listing of the filenames in directory.
ls -l	Long List	Show a comprehensive directory list.
ls /bin	List /bin Directory	Show a list of valid commands.
cd	Change Directory	Change working directory.
chmod	Change Permissions	Change permissions on a file or directory.
cp	Copy	Copy a file into another file or directory.
mv	Move	Move a file or directory.
more	Show More	Type the file's contents to the screen.
lpr	Print	Print out a file.
date	Date	Show date and time.
mkdir	Make Directory	Make a new directory.
grep	Global Regular Expression/Print	Find a specified string in a file.
cat	Concatenate or Catenate	Concatenate the files and print the resulting file.
diff	Different	Compare two files.
pwd	Print Working Directory	Print the name of the working directory.

Linux: Command Line Interface

http://www.masswerk.at/jsuix/

```
ls
ls -la
pwd
cd .
cd ..
man man
```

Linux: Command Line Interface

http://www.masswerk.at/jsuix/

```
#!/bin/sh
mkdir BackUpFolder
cp *.txt BackUpFolder
ls -la BackUpFolder
```