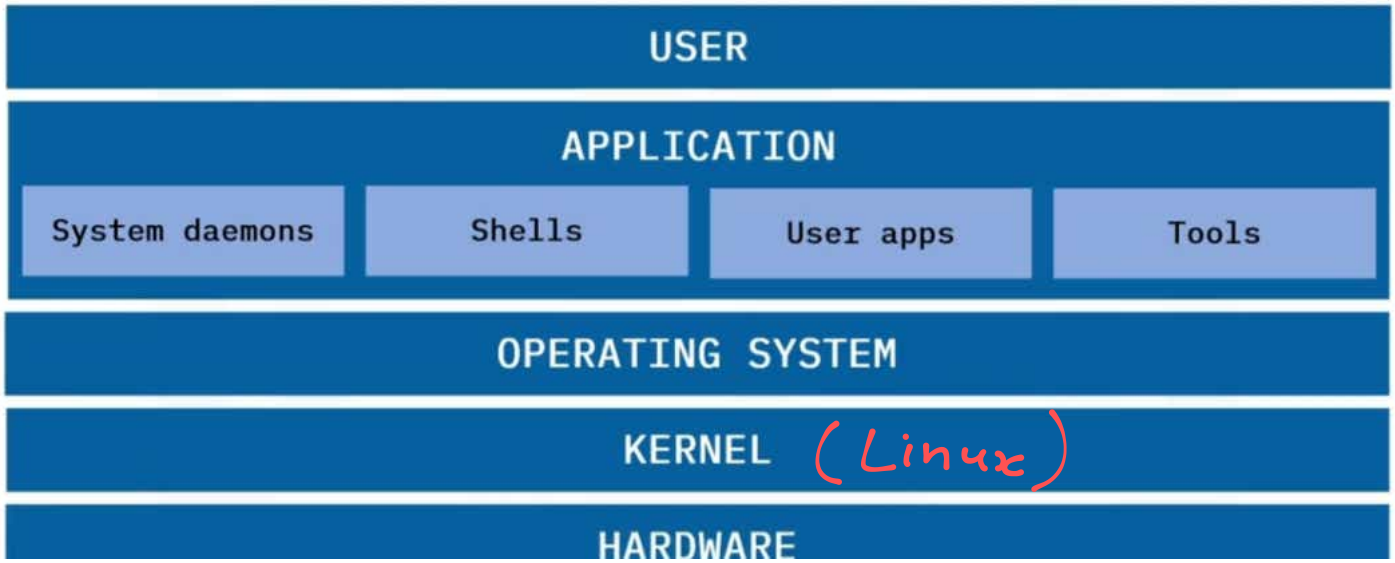
→ Developer
→ DevOps

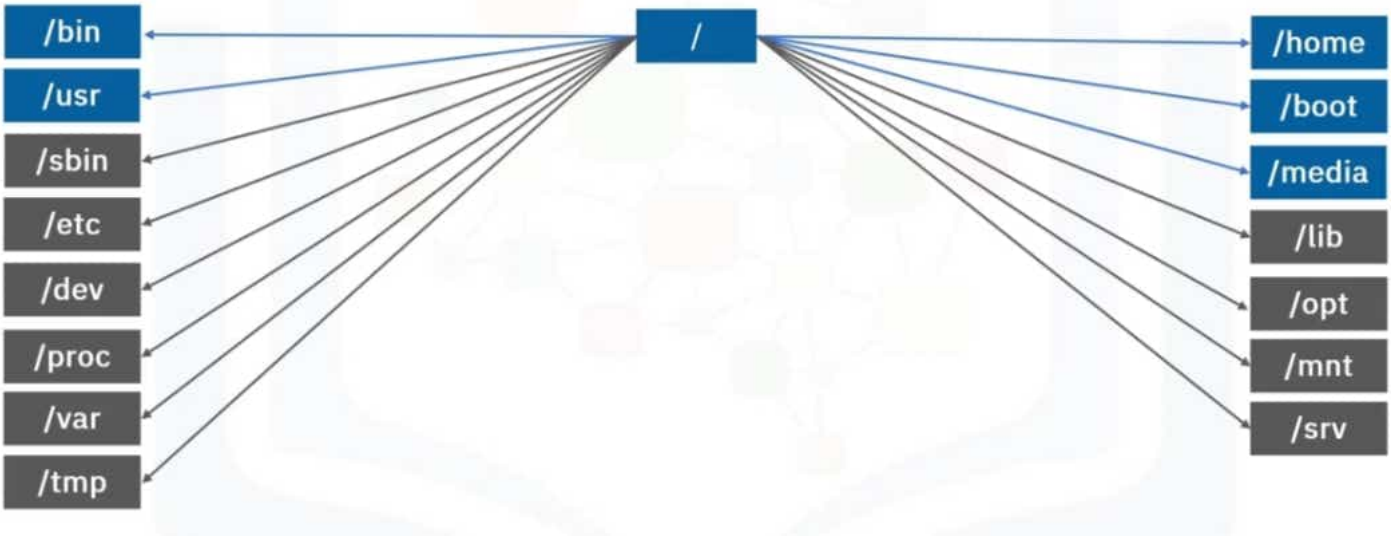
Linux beginnings

```
Date: 25 Aug 91 20:57:08 GMT Organization: University of Helsinki

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID:
Date: 25 Aug 91 20:57:08 GMT Organization: University of Helsinki

Hello everybody out there using minix - I'm doing a (free) operating system (just
a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has
been brewing since april, and is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat (same
physical layout of the file-system (due to practical reasons) among other
things).
```

Overview


USER
APPLICATION
System daemons
Shells
User apps
Tools
OPERATING SYSTEM
KERNEL (Linux)
HARDWARE

Linux filesystem

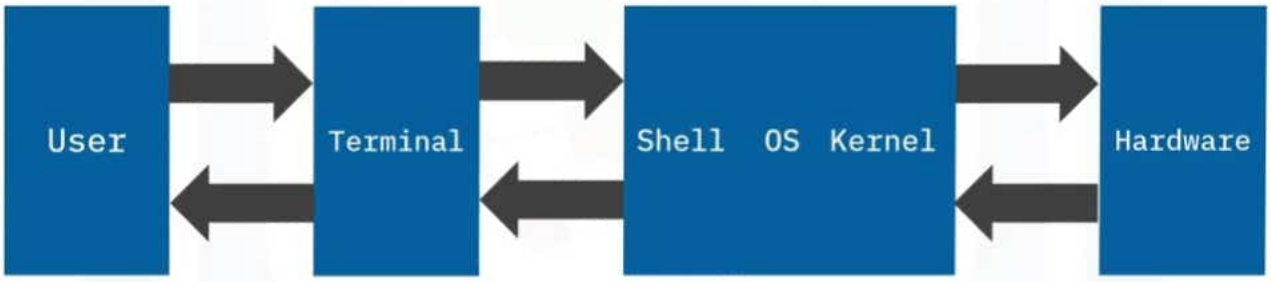


Uses (Linux Kernel)

Linux distros: Ubuntu

- First release in 2004 (4.10)
- Debian-based
- Developed and managed by Canonical
- Three editions:
  - Ubuntu Desktop
  - Ubuntu Server
  - Ubuntu Core (for IOT)

Communicating with Linux system



Paths in the Linux filesystem

- Human-readable directory or file location

  /home/me/Documents/

- a/b - the file or directory named b inside the directory named a
- Special paths:

  ~ Home directory
  / Root directory
  .. Parent of current directory
  . Current directory

cd ~

Changing the current working directory

```
/home $ cd ..
/ $ cd /media/my-usb-drive
/media/my-usb-drive $ cd ../../home/me/Documents
/home/me/Documents $
```

DIY

Popular text editors

- Command-line text editors:
  - GNU nano
  - vi
  - vim
- GUI-based text editors:
  - gedit
- Command-line or GUI:
  - emacs

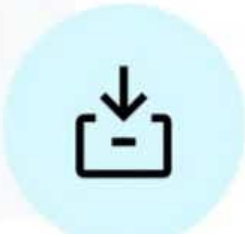
Deb and RPM packages

- deb and RPM formats are equivalent
- If a package is only available in one format you can use alien to convert it:
  - RPM to deb

```
alien <package-name>.rpm
```

  - deb to RPM

```
alien -r <package-name>.deb
```



Installing new software

Installing a deb package with apt:

```
sudo apt install <package-name>
```

Installing an RPM package with yum:

```
sudo yum install <package-name>
```

Other software package managers

- Python package managers include pip and conda
- Installing the pandas library:

```
pip install pandas

Collecting pandas
Downloading pandas-1.4.1-cp38-cp38-manylinux1_x86_64.whl (10.3 MB)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/lib/python3/dist-packages (from pandas) (2.7.3)
Requirement already satisfied: pytz>=2017.2 in /usr/lib/python3/dist-
packages (from pandas) (2019.3)
Requirement already satisfied: numpy>=1.15.4 in /usr/lib/python3/dist-
packages (from pandas) (1.17.4)
Installing collected packages: pandas
```

What is a shell?

- User interface for running commands

- Interactive language

- Scripting language

A sea of shells

- Default shell is usually Bash
- Many other shells, including sh, ksh, tcsh, zsh, and fish
- We will use Bash for this course

```
$ printenv SHELL
/bin/bash
$ bash
```

```
$
```

```
>
```

Getting information

Some common shell commands for getting information include:

- whoami – username
- id – user ID and group ID
- uname – operating system name
- ps – running processes
- top – resource usage
- df – mounted file systems
- man – reference manual
- date – today's date

Working with files

Some common shell commands for working with files include:

- cp – copy file
- mv – change file name or path
- rm – remove file
- touch – create empty file, update file timestamp
- chmod – change/modify file permissions
- wc – get count of lines, words, characters in file
- grep – return lines in file matching pattern

Navigating & working with directories

Very common shell commands for navigating and working with directories include:

- ls – list files and directories
- find – find files in directory tree
- pwd – get present working directory
- mkdir – make directory
- cd – change directory
- rmdir – remove directory

Printing file and string contents

For printing file contents or strings, common commands include:

- cat – print file contents
- more – print file contents page-by-page
- head – print first N lines of file
- tail – print last N lines of file
- echo – print string or variable value

Compression and archiving

Shell commands related to file compression and archiving applications include:

- tar – archive a set of files
- zip – compress a set of files
- unzip – extract files from a compressed zip archive

Networking

Networking applications include the following:

- hostname – print hostname
- ping – send packets to URL and print response
- ifconfig – display or configure system network interfaces
- curl – display contents of file at a URL
- wget – download file from URL

Running Linux on a Windows machine

- Dual boot with a partition
- Install Linux on a virtual machine
- Use a Linux emulator (Putty)
- Windows Subsystem for Linux (WSL)

User Information

- `whoami` – returns user ID
- `id` (Identity) – user ID or group ID

```
$ whoami
johndoe

$ id -u
501
$ id -u -n
johndoe
```

System Information

uname (Unix Name) - returns OS Information

- Kernel/os name, version number

```
$ uname
Darwin
$ uname -s -r
Darwin 20.6.0
$ uname -v
Darwin Kernel Version 20.6.0: Mon Aug 30 06:12:21 PDT 2021;
root:xnu-7195.141.6~3/RELEASE_X86_64
```

Displaying your disk usage

df (Disk Free) - Shows disk usage

```
$ df -h ~
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme0n1p2  2.0T  744G  1.2T  40% /home
```

Getting disk usage information

df (Disk Free) – Shows disk usage

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev             26G     0   26G   0% /dev
tmpfs           5.1G  2.6M  5.1G   1% /run
/dev/nvme1n1p5  255G   65G  177G  27% /
tmpfs            26G  223M   25G   1% /dev/shm
tmpfs           5.3M  4.1k  5.3M   1% /run/lock
tmpfs            26G     0   26G   0% /sys/fs/cgroup
/dev/loop2      230M  230M     0 100% /snap/gnome-3-34-1804/66
/dev/loop0      132k  132k     0 100% /snap/bare/5
/dev/loop1       59M   59M     0 100% /snap/core18/2128
…
```

Displaying current running processes

- ps (Process status) – Running processes

```
$ ps -u root
 UID   PID TTY           TIME CMD
   0     1 ??         8:15.69 /sbin/launchd
   0    76 ??         0:13.27 /usr/sbin/syslogd
…
```

Monitoring system health and status

`top` (Table of Processes) – Task manager

- Shows running tasks and their resource usage

```
$ top -n 3
PID    COMMAND          %CPU TIME      ...  USER      ...
38702  chrome           10.0 01:00.41  ...  johndoe   ...
38701  top              4.0  00:00.09  ...  johndoe   ...
38699  Spotify          3.0  01:00.07  ...  johndoe   ...
```

Getting variable values

echo – Print string or variable value

```
$ echo

$ echo hello
hello
$ echo "Learning Linux is fun!"
Learning Linux is fun!
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

Getting date information

date – Displays system date and time

```
$ date
Thu 16 Sep 2021 16:50:49 EDT
$ date "+%j day of %Y"
259 day of 2021
$ date "+It's %A, the %j day of %Y!"
It's Thursday, the 259 day of 2021!
```

Miscellaneous Information

- man (Manual) – Shows manual for any command

```
$ man id
NAME
     id -- return user identity

SYNOPSIS
     id [user]
     id -A
     id -F [user]
…

DESCRIPTION
     The id utility displays the user and group names and numeric
IDs, of the calling process, to the standard output.  If the real…
```

Listing your directory contents

`ls` (list) - list files and directories

```
$ ls
Documents Downloads Music Pictures
$ ls Downloads
download1.zip
download2.zip
download3.zip
```

Listing your directory contents

ls (list) - list files and directories

```
$ pwd
/Users/me/Documents
$ ls -l
-rwxr-xr-x me staff 21 Sep 06:45 assignment-1.txt
-rwxr-xr-x me staff 09 Feb 03:27 assignment-2.txt
-rwxr-xr-x me staff  1 Jan 01:23 notes-1.txt
-rwxr-xr-x me staff  3 Aug 10:03 notes-2.txt
-rwxr-xr-x me staff  7 Nov 16:21 notes-3.txt
-rwxr-xr-x me staff 27 Sep 04:56 notes-4.txt
```

Navigating your directories

cd (change directory) – change directory

```
$ pwd
/Users/me
$ ls
Documents Downloads Music Pictures
$ cd Documents
$ pwd
/Users/me/Documents
```

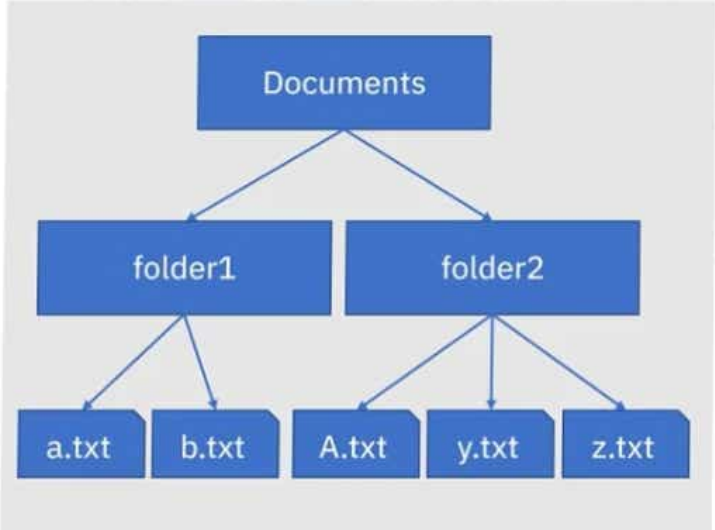
Relative and absolute navigation

cd (change directory) – change directory

```
$ pwd
/Users/me/Documents/Academics/Math/Notes
$ cd ..
$ pwd
/Users/me/Documents/Academics/Math
$ cd ~
$ pwd
/Users/me
$ cd /Users/me/Documents/Academics/Math/Notes
```

Finding files

find – find files in directory tree




```
$ pwd
/Users/me/Documents
$ find . -name "a.txt"
./folder1/a.txt
$ find . -iname "a.txt"
./folder1/a.txt
./folder2/A.txt
```

File and Directory Management Commands

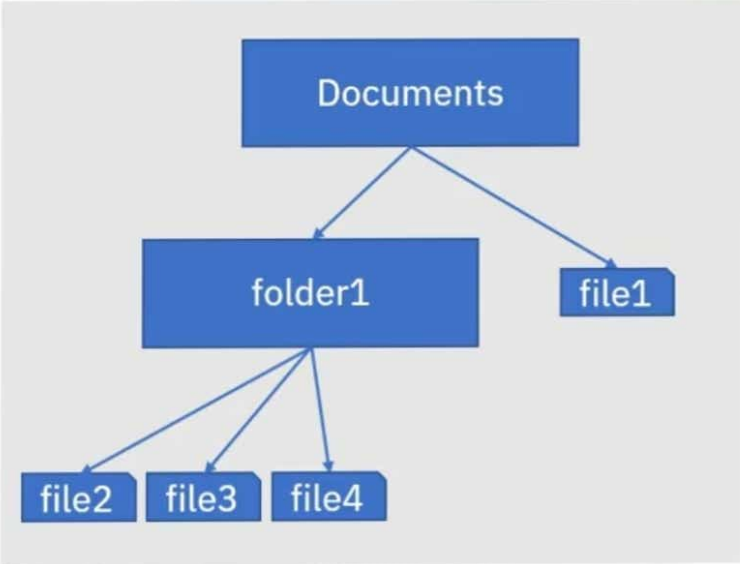
Creating directories

`mkdir` (make directory) – make directory

```
$ pwd
/Users/me/Documents
$ ls

$ mkdir test
$ ls
test
```

Removing files and directories

`rm` (remove) – Remove file or directory




```
$ pwd
/Users/me/Documents
$ ls
file1 folder1
$ rm file1
$ ls
folder1
$ rm folder1
rm: folder1: is a directory
$ rm -r folder1
$ ls

$ mkdir empty_folder
$ rmdir empty_folder
```

Creating files

touch – Create empty file, update file date

```
$ pwd
/Users/me/Documents
$ touch a.txt b.txt c.txt d.txt
$ ls
a.txt b.txt c.txt d.txt
$ date -r notes.txt
Mon  8 Nov 2021 16:37:45 EST
$ touch notes.txt
$ date -r notes.txt
Fri 12 Nov 2021 10:46:03 EST
```

Copying files and directories

cp (copy) - Copy file or directory to destination

To copy files:

```
$ cp /source/file /dest/filename
$ cp /source/file /dest/
```

To copy directories:

```
$ cp -r /source/dir/ /dest/dir/
```

```
$ ls
notes.txt Documents
$ cp notes.txt Documents
$ ls Documents
notes.txt
$ cp -r Documents Docs_copy
$ ls
notes.txt Documents Docs_copy
$ ls Docs_copy
notes.txt
```

Moving files and directories

mv (move) - Move a file or directory

To move files:

```
$ mv /source/file /dest/dir/
```

To move directories:

```
$ mv /source/dir/ /dest/dir/
```

```
$ ls
my_script.sh Scripts Notes Documents
$ mv my_script.sh Scripts
$ ls my_script.sh

$ ls Scripts
my_script.sh
$ mv Notes Scripts Documents
$ ls
Documents
```

Managing file permissions

chmod (change mode) – Change file permissions

```
$ ls -l my_script.sh
-rw-r--r— my_script.sh
$ ./my_script.sh
bash: permission denied: ./my_script.sh
$ chmod +x my_script.sh
$ ls -l my_script.sh
-rwxr-xr-x my_script.sh
$ ./my_script.sh
Learning Linux is fun!
```

Viewing your file all at once

cat ("catenate") - Print entire file contents

```
$ ls
numbers.txt
$ cat numbers.txt
```

```
89
90
91
92
93
94
95
96
97
98
99
```

Viewing your file page-by-page

more – Print file contents page-by-page

```
$ more numbers.txt
```

space bar

```
9
10
11
12
13
14
15
16
17
```

Viewing your file page-by-page

more – Print file contents page-by-page

```
$ more numbers.txt
```

q + enter

```
$
```

Viewing the first 10 lines

head - Print first 10 lines of file

```
$ head numbers.txt
0
1
2
3
4
5
6
7
8
9
```

Viewing the first N lines

head - Print first N lines of file

```
$ head -n 3 numbers.txt
0
1
2
```

Viewing the last 10 lines

tail – Print last 10 lines of file

```
$ tail numbers.txt
90
91
92
93
94
95
96
97
98
99
```

Viewing the last N lines

`tail` – Print last N lines of file

```
$ tail -n 3 numbers.txt
97
98
99
```

Counting lines, words, and characters

wc (word count) – Count characters, words, lines

```
$ cat pets.txt
cat
cat
cat
cat
dog
dog
cat
```

```
$ wc pets.txt
7  7  28 pets.txt
```

Counting lines, words, and characters

wc (word count) – Count characters, words, lines

```
$ cat pets.txt
cat
cat
cat
cat
dog
dog
cat
```

```
$ wc pets.txt
7  7  28 pets.txt
$ wc -l pets.txt
7  pets.txt
$ wc -w pets.txt
7  pets.txt
$ wc -c pets.txt
28 pets.txt
```

Sorting your views line-by-line

sort - Sort lines in a file

```
$ sort pets.txt
cat
cat
cat
cat
cat
dog
dog
```

```
$ sort -r pets.txt
dog
dog
cat
cat
cat
cat
cat
```

Excluding repeated lines from views

uniq ("unique") - Filter out repeated lines

```
$ cat pets.txt
cat
cat
cat
cat
dog
dog
cat
```

```
$ uniq pets.txt
cat
dog
cat
```

Extracting lines matching a pattern

grep ("global regular expression print") – Return lines in file matching pattern

```
$ cat people.txt
Alan Turing
Bjarne Stroustrup
Charles Babbage
Dennis Ritchie
Erwin Schrodinger
Fred Hoyle
Guido Rossum
Henri Poincare
Ivan Pavlov
John Neumann
Ken Thompson
```

```
$ grep ch people.txt
Dennis Ritchie
Erwin Schrodinger
```

Extracting lines matching a pattern

grep ("global regular expression print") – Return lines in file matching pattern

```
$ cat people.txt
Alan Turing
Bjarne Stroustrup
Charles Babbage
Dennis Ritchie
Erwin Schrodinger
Fred Hoyle
Guido Rossum
Henri Poincare
Ivan Pavlov
John Neumann
Ken Thompson
```

```
$ grep ch people.txt
Dennis Ritchie
Erwin Schrodinger

$ grep -i ch people.txt
Charles Babbage
Dennis Ritchie
Erwin Schrodinger
```

Extracting slices from lines

cut – Extracts a section from each line

```
$ cat people.txt
Alan Turing
Bjarne Stroustrup
Charles Babbage
Dennis Ritchie
Erwin Schrodinger
Fred Hoyle
Guido Rossum
Henri Poincare
Ivan Pavlov
John Neumann
Ken Thompson
```

```
$ cut -c 2-9 people.txt
lan Turi
jarne St
harles B
ennis Ri
rwin Sch
red Hoyl
uido Ros
enri Poi
van Pavl
ohn Neum
en Thomp
```

Extracting fields from lines

cut – Extract a field from each line

```
$ cat people.txt
Alan Turing
Bjarne Stroustrup
Charles Babbage
Dennis Ritchie
Erwin Schrodinger
Fred Hoyle
Guido Rossum
Henri Poincare
Ivan Pavlov
John Neumann
Ken Thompson
```

```
$ cut -d ' ' -f2 people.txt
Turing
Stroustrup
Babbage
Ritchie
Schrodinger
Hoyle
Rossum
Poincare
Pavlov
Neumann
Thompson
```

Merging lines from multiple files

paste – Merge lines from different files

```
$ paste first.txt last.txt yob.txt
Alan    Turing       1912
Bjarne  Stroustrup   1950
Charles Babbage      1791
Dennis  Ritchie      1941
Erwin   Schrodinger  1887
Fred    Hoyle        1915
Guido   Rossum       1956
Henri   Poincare     1854
Ivan    Pavlov       1849
John    Neumann      1903
Ken     Thompson     1943
```

Merging lines from multiple files

paste – Merge of lines from different files

```
$ paste -d "," first.txt last.txt yob.txt
Alan,Turing,1912
Bjarne,Stroustrup,1950
Charles,Babbage,1791
Dennis,Ritchie,1941
Erwin,Schrodinger,1887
Fred,Hoyle,1915
Guido,Rossum,1956
Henri,Poincare,1854
Ivan,Pavlov,1849
John,Neumann,1903
Ken,Thompson,1943
```

Archiving and compression

Archives:

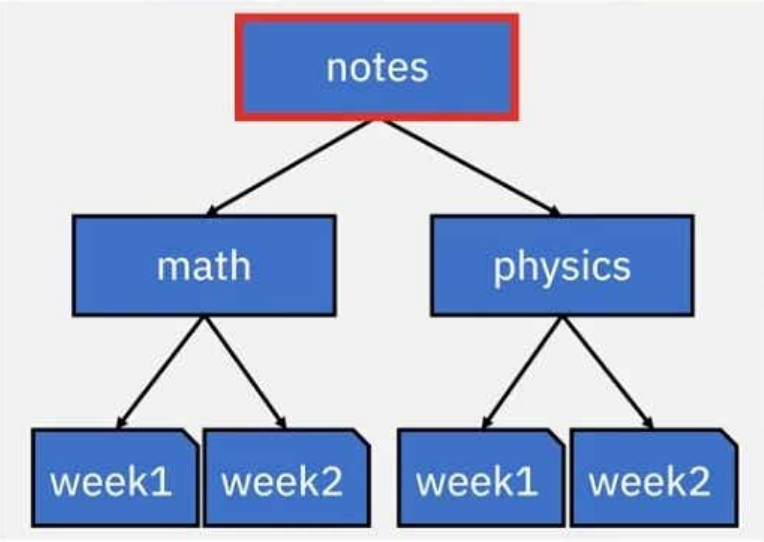
- Store rarely used information and preserve it
- Are a collection of data files and directories stored as a single file
- Make the collection more portable and serve as a backup in case of loss or corruption

File compression:

- Reduces file size by reducing information redundancy
- Preserves storage space, speeds up data transfer, and reduces bandwidth load

Directory tree archiving

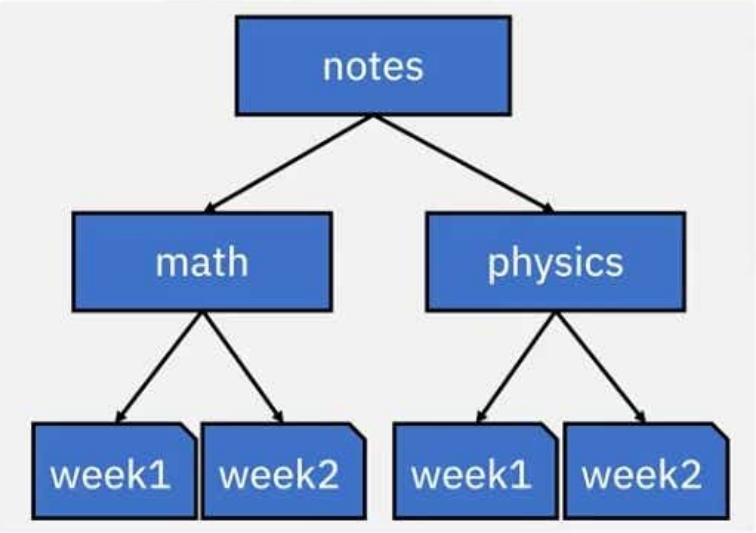
Notes directory tree example




```
$ ls -R
.:
notes
./notes:
math  physics
./notes/math:
week1  week2
./notes/physics:
week1  week2
```

File archiving and compression

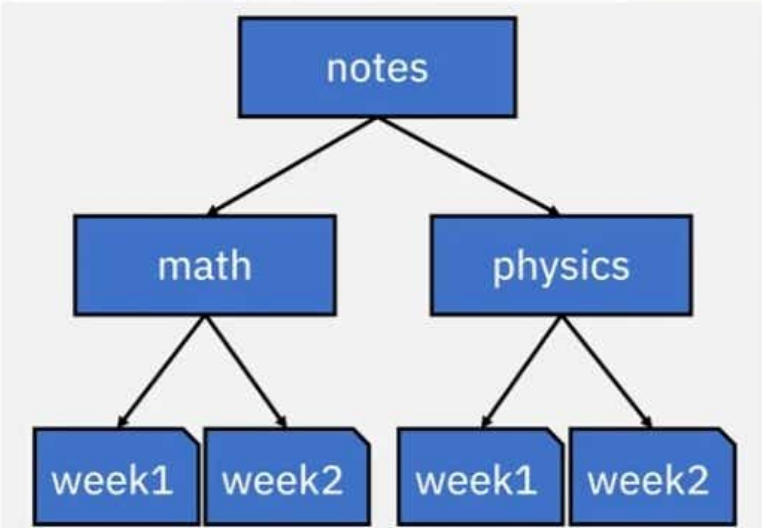
`tar` (tape archiver) – Archive and extract files




```
$ tar -cf notes.tar notes
$ ls
notes notes.tar
$ tar -czf notes.tar.gz notes
$ ls
notes notes.tar notes.tar.gz
```

Checking your archive contents

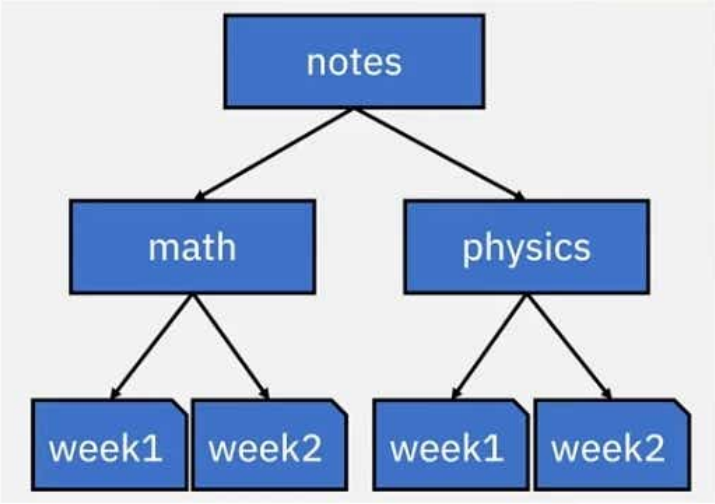
`tar` – List archive contents




```
$ tar -tf notes.tar
notes/
notes/math/
notes/physics/
notes/physics/week1
notes/physics/week2
notes/math/week1
notes/math/week2
```

Extracting archived files

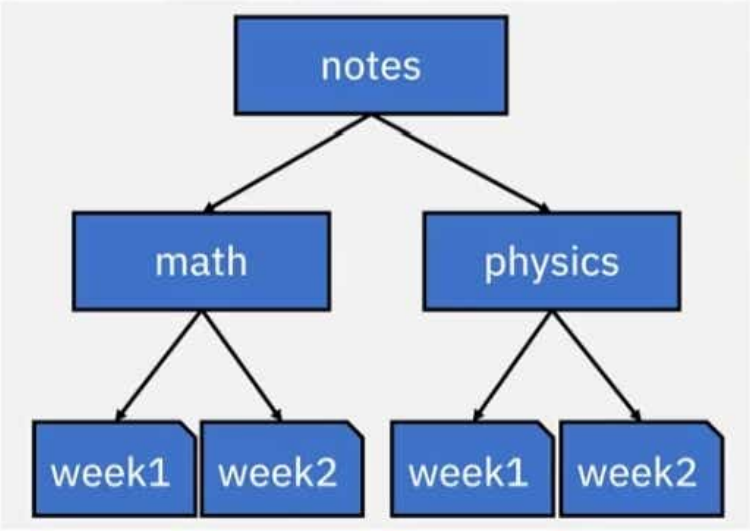
`tar` – Extract files and folders




```
$ tar -xf notes.tar notes
$ ls -R
.:
notes  notes.tar
./notes:
math  physics
./notes/math:
week1  week2
./notes/physics:
week1  week2
```

Decompressing and extracting archives

tar – Decompress and extract




```
$ tar -xzf notes.tar.gz notes
$ ls -R
.:
notes  notes.tar
./notes:
math  physics
./notes/math:
week1  week2
./notes/physics:
week1  week2
```

File compression and archiving

`zip` – Compress files and directories to an archive

`zip`:

compress → bundle

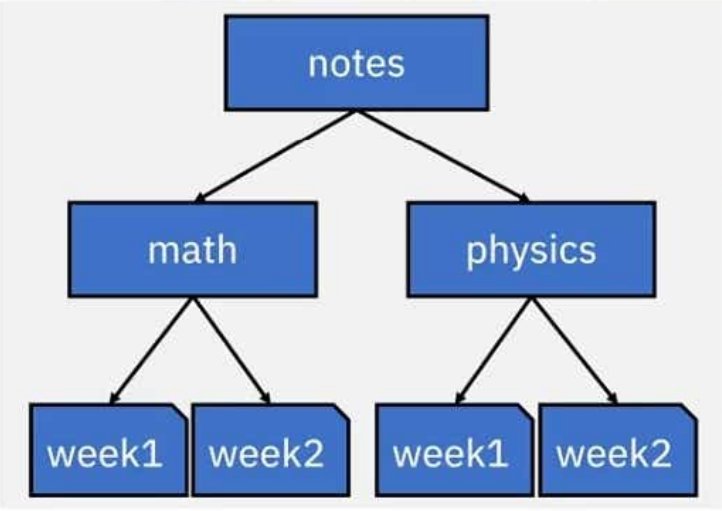
`tar`:

bundle → compress

```
$ zip notes.zip notes
$ ls
notes notes.zip
```

Extracting and decompressing archives

`unzip` – Extract and decompress zipped archive




```
$ unzip notes.zip
$ ls -R
.:
notes  notes.tar
./notes:
math  physics
./notes/math:
week1  week2
./notes/physics:
week1  week2
```

Getting your machine's host name

- `hostname` – Print host name

```
$ hostname
my-linux-machine.local
$ hostname -s
my-linux-machine
$ hostname -i
127.0.1.1
```

Getting network information

ifconfig (Interface configuration) – Display or configure the system network interfaces

```
$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
  options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
  inet 127.0.0.1 netmask 0xff000000
  inet6 ::1 prefixlen 128
  inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
  nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
…
```

Getting ethernet adapter info

ifconfig (Interface configuration) - Display or configure system network interfaces

```
$ ifconfig eth0
eth0: Link encap:Ethernet  HWaddr 00:0B:CD:1C:18:5A
  inet addr:172.16.25.126  Bcast:172.16.25.63  Mask:255.255.255.224
  inet6 addr: fe80::20b:cdff:fe1c:185a/64 Scope:Link
  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
  RX packets:2345583 errors:0 dropped:0 overruns:0 frame:0
  TX packets:2221421 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:293912265 (280.2 MiB)  TX bytes:1044100408 (995.7 MiB)
  Interrupt:185 Memory:f7fe0000-f7ff0000
```

Testing server connections

`ping` – Send ICMP packets to URL and print response

```
$ ping www.google.com
PING www.google.com (142.251.41.68): 56 data bytes
64 bytes from 142.251.41.68: icmp_seq=0 ttl=119 time=21.750 ms
64 bytes from 142.251.41.68: icmp_seq=1 ttl=119 time=20.712 ms
64 bytes from 142.251.41.68: icmp_seq=2 ttl=119 time=24.065 ms
64 bytes from 142.251.41.68: icmp_seq=3 ttl=119 time=36.751 ms
64 bytes from 142.251.41.68: icmp_seq=4 ttl=119 time=41.774 ms
^C
--- www.google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.712/229.010/41.774/9.603 ms
```

Testing server connections

`ping` – Send ICMP packets to URL and print response

```
$ ping -c 5 www.google.com
PING www.google.com (142.251.41.68): 56 data bytes
64 bytes from 142.251.41.68: icmp_seq=0 ttl=119 time=17.491 ms
64 bytes from 142.251.41.68: icmp_seq=1 ttl=119 time=19.784 ms
64 bytes from 142.251.41.68: icmp_seq=2 ttl=119 time=24.279 ms
64 bytes from 142.251.41.68: icmp_seq=3 ttl=119 time=24.964 ms
64 bytes from 142.251.41.68: icmp_seq=4 ttl=119 time=26.106 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 17.491/22.525/26.106/3.308 ms
```

Web scraping with curl

`curl` (Client URL) – Transfer data to and from URL(s)

```
$ curl www.google.com
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"
lang="en-CA"><head><meta content="text/html; charset=UTF-8" http-
equiv="Content-Type"><meta
content="/images/branding/googleg/1x/googleg_standard_color_128dp.png"
itemprop="image"><title>Google</title><script
nonce="gPa6M7RHuxLHFwYnP5CH4A==">(function(){window.google={kEI:'FdCKYdj-
LrOt0PEPuoqLIA',kEXPI:'0,18168,1284368,56873,1709,4350,206,4804,2316,383,
246,5,1354,5250,1122516,1197719,329548,51224,16114,17444,11240,17572,4859
,1361,9291,3027,2816,1931,12834,4020,978,13228,516,3331,4192,6430,7432,14
390,919,5081,887,706,1279,2212,530,149,1103,840,1983,213,4101,3514,606,20
...
```

Scraping a web page's HTML to file

curl (Client URL) – Transfer data to and from URL(s)

```
$ curl www.google.com -o google.txt
$ head -n 1 google.txt
<!doctype html><html itemscope=""
itemtype="http://schema.org/WebPage" lang="en-CA"><head><meta
content="text/html; charset=UTF-8" http-equiv="Content-
Type"><meta content="/images/branding/googleg/1x/googleg_standard_color_1
28dp.png" itemprop="image"><title>Google</title><script nonce="gPa6M7RHux
LHFwYnP5CH4A==">(function(){window.google={kEI:'FdCKYdj-
LrOt0PEPuoqLIA',kEXPI:'0,18168,1284368,56873,1709,4350,206,4804,2316,383,
246,5,1354,5250,1122516,1197719,329548,51224,16114,17444,11240,17572,4859
...
```

Downloading files from a URL

`wget` (Web get) – Download file(s) from a URL

- more focused than `curl`, supports recursive file downloads

```
$ wget https://www.w3.org/TR/PNG/iso_8859-1.txt
Resolving www.w3.org (www.w3.org)... 128.30.52.100
Connecting to www.w3.org (www.w3.org)|128.30.52.100|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6121 (6.0K) [text/plain]
Saving to: ‘iso_8859-1.txt’

iso_8859-
1.txt                          100%[===================================>]
  5.98K  --.-KB/s    in 0s
```

Downloading files from a URL

https://www.w3.org/TR/

w3.org/TR/PNG/iso_8859-1.txt

```
The following are the graphical (non-control) characters defined by
ISO 8859-1 (1987).  Descriptions in words aren't all that helpful,
but they're the best we can do in text.  A graphics file illustrating
the character set should be available from the same archive as this
file.

Hex Description                 Hex Description

20  SPACE
21  EXCLAMATION MARK            A1  INVERTED EXCLAMATION MARK
22  QUOTATION MARK              A2  CENT SIGN
23  NUMBER SIGN                 A3  POUND SIGN
24  DOLLAR SIGN                 A4  CURRENCY SIGN
25  PERCENT SIGN                A5  YEN SIGN
26  AMPERSAND                   A6  BROKEN BAR
27  APOSTROPHE                  A7  SECTION SIGN
28  LEFT PARENTHESIS            A8  DIAERESIS
29  RIGHT PARENTHESIS           A9  COPYRIGHT SIGN
2A  ASTERISK                    AA  FEMININE ORDINAL INDICATOR
```

Downloading files from a URL

```
$ head -n 12 iso_8859-1.txt
The following are the graphical (non-control) characters defined by
ISO 8859-1 (1987).  Descriptions in words aren't all that helpful,
but they're the best we can do in text.  A graphics file illustrating
the character set should be available from the same archive as this
file.

Hex Description                    Hex Description

20  SPACE
21  EXCLAMATION MARK               A1  INVERTED EXCLAMATION MARK
22  QUOTATION MARK                 A2  CENT SIGN
23  NUMBER SIGN                    A3  POUND SIGN
```

Shell Scripting Basics

IBM Developer

SKILLS NETWORK

What is a script?

- Script: list of commands interpreted by a scripting language
- Commands can be entered interactively or listed in a text file
- Scripting languages are interpreted at runtime
- Scripting is slower to run, but faster to develop

What is a script used for?

- Widely used to automate processes
- ETL jobs, file backups and archiving, system admin
- Used for application integration, plug-in development, web apps, and many other tasks

Shell scripts and the ‘shebang’

- Shell script – executable text file with an *interpreter directive*
- Aka ‘shebang’ directive:

```
#!interpreter [optional-arg]
```

- ‘*interpreter*’ – path to an executable program

Shell scripts and the 'shebang'

- Shell script – executable text file with an *interpreter directive*
- Aka 'shebang' directive:

```
#!interpreter [optional-arg]
```

- *'interpreter'* – path to an executable program
- *'optional-arg'* – single argument string

Example – ‘shebang’ directives

Shell script directives:

```
#!/bin/sh
#!/bin/bash
```

Python script directive:

```
#!/usr/bin/env python3
```

‘Hello World’ example shell script

Create the shell script:

```
$ touch hello_world.sh
$ echo '#! /bin/bash' >> hello_world.sh
```

‘Hello World’ example shell script

Create the shell script:

```
$ touch hello_world.sh
$ echo '#! /bin/bash' >> hello_world.sh
$ echo 'echo hello world' >> hello_world.sh
```

‘Hello World’ example shell script

Make it executable:

```
$ ls -l hello_world.sh
-rw-rw-r-- 1 jgrom jgrom 12 Jun 27 09:13 hello_world.sh
$ chmod +x hello_world.sh
$ ls -l hello_world.sh
-rwxrwxr-x 1 jgrom jgrom 12 Jun 27 09:13 hello_world.sh
```

‘Hello World’ example shell script

Run your bash script:

```
$ ./hello_world.sh
hello world
```

Filters, Pipes and Variables

Pipes and filters

Filters are shell commands, which:

- Take input from standard input
- Send output to standard output
- Transform input data into output data
- Examples are wc, cat, more, head, sort, …
- Filters can be chained together

Pipes and filters

Pipe command – |

- For chaining filter commands

```
command1 | command2
```

- Output of command 1 is input of command 2

Pipes and filters

Pipe command – |

- For chaining filter commands

```
command1 | command2
```

- Output of command 1 is input of command 2
- Pipe stands for pipeline

```
$ ls | sort -r
Videos
Public
Pictures
Music
Downloads
Documents
Desktop
```

Shell variables

- Scope limited to shell
- Set – list all shell variables

```
$ set | head -4
BASH=/usr/bin/bash
BASHOPTS=checkwinsize:cm
  dhist:complete_fullquot
  e:expand_aliases:extglo
  b:extquote:force_fignor
  e:globasciiranges:hista
  ppend:interactive_comme
  nts:progcomp:promptvars
  :sourcepath
BASH_ALIASES=()
BASH_ARGC=([0]="0")
```

Defining shell variables

```
var_name=value
```

- No spaces around '='

```
unset var_name
```

- deletes var_name

```
$ GREETINGS="Hello"
$ echo $GREETINGS
Hello

$ AUDIENCE='World'
$ echo $GREETINGS $AUDIENCE
Hello World

$ unset AUDIENCE
```

Environment variables

- Extended scope

```
export var_name
```

- env – list all environment variables

```
$ export GREETINGS

$ env | grep "GREE"
$ GREETINGS=Hello
```

Metacharacters

- precedes a comment

; - command separator

* - filename expansion wildcard

```
$ # Some metacharacters

$ echo "Hello"; whoami
Hello
ravahuja

$ ls /bin/ba*
/bin/bash
```

Metacharacters

- precedes a comment

; - command separator

* - filename expansion wildcard

? - single character wildcard in filename expansion

```
$ # Some metacharacters

$ echo "Hello"; whoami
Hello
ravahuja

$ ls /bin/ba*
/bin/bash
$ ls /bin/?ash
/bin/bash /bin/dash
```

Quoting

\ – escape special character interpretation

" " – interpret literally, but evaluate metacharacters

```
$ echo "\$1 each"
$1 each

$ echo "$1 each"
 each
```

Quoting

\ – escape special character interpretation

" " – interpret literally, but evaluate metacharacters

' ' – interpret literally

```
$ echo "\$1 each"
$1 each

$ echo "$1 each"
 each


$ echo '$1 each'
$1 each
```

I/O redirection

Input/Output, or I/O redirection, refers to a set of features used for redirecting

```
>    - Redirect output to file
>>   - Append output to file
2>   - Redirect standard error to file
2>>  - Append standard error to file
<    - Redirect file contents to standard input
```

I/O redirection examples

```
$ echo "line1" > eg.txt
$ cat eg.txt
line1
$ echo "line2" >> eg.txt
$ cat eg.txt
line1
line2
```

```
$ garbage
garbage: command
not found
$ garbage 2> err.txt
$ cat err.txt
garbage: command not
found
```

Command substitution

- Replace command with its output
  $(command) or `command`
- Store output of 'pwd' command in 'here':

```
$ here=$(pwd)
$ echo $here
/home/jgrom
```

Command line arguments

- Program arguments specified on the command line
- A way to pass arguments to a shell script
- Usage:

```
$ ./MyBashScript.sh arg1 arg2
```

Batch vs. concurrent modes

Batch mode:

- Commands run sequentially

```
command1; command2
```

Concurrent mode:

- Commands run in parallel

```
command1 & command2
```

Scheduling Jobs Using Cron

Job scheduling

- Schedule jobs to run automatically at certain times

Load script at midnight every night
Backup script to run every Sunday at 2 AM

- Cron allows you to automate such tasks

What are cron, crond, and crontab?

- Cron is a service that runs jobs
- Crond interprets ‘crontab files’ and submits jobs to cron
- A crontab is a table of jobs and schedule data
- Crontab command invokes text editor to edit a crontab file

Scheduling cron jobs with crontab

```
$ crontab -e    # opens editor
```

Job syntax:

```
m h dom mon dow command
                -------
```

Example job:

```
30 15 * * 0 date >> sundays.txt
```

Scheduling cron jobs with crontab

```
  GNU nano 4.8                    /tmp/crontab.8We1DK/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Entering jobs

```
# m  h   dom mon dow  command

 30 15  *   *   0    date >> path/sundays.txt
  0  0  *   *   *    /cron_scripts/load_data.sh
  0  2  *   *   0    /cron_scripts/backup_data.sh
```

Exit editor and save

```
# m  h   dom mon dow  command

  30 15  *   *   0    date >> path/sundays.txt
   0  0  *   *   *    /cron_scripts/load_data.sh
   0  2  *   *   0    /cron_scripts/backup_data.sh

Save modified buffer?
 Y Yes
 N No            ^C Cancel
```

Viewing and removing cron jobs

```
jgrom@GROOT617:~$ crontab -l  | tail -6
#
# m  h   dom mon dow  command

  30 15  *   *   0    date >> path/sundays.txt
   0  0  *   *   *    /cron_scripts/load_data.sh
   0  2  *   *   0    /cron_scripts/backup_data.sh
jgrom@GROOT617:~$
```

Viewing and removing cron jobs

```
jgrom@GROOT617:~$ crontab -l  | tail -6
#
# m  h   dom mon dow  command

  30 15  *   *   0    date >> path/sundays.txt
   0  0  *   *   *    /cron_scripts/load_data.sh
   0  2  *   *   0    /cron_scripts/backup_data.sh
jgrom@GROOT617:~$
```

```
$ crontab -e  # add/remove cron job with editor
```