Contents

Commands taught over the weeks - System Commands

Week 1

Launching a Linux Virtual Machine

link

- 00:00 `VirtualBox`
- 15:00 `WSL`
- 18:50 `Replit`
- 21:36 `CoCalc`
- 25:00 `Termux`

Command Line Environment

link

- 02:05 `pwd`
- 03:22 `ls`
- 03:49 `ps`
- 04:06 `uname`
- 04:15 `clear or ctrl-L`
- 04:35 `exit or ctrl-D`
- 06:20 `anatomy of command`
- 07:29 `ls -a`
- 08:13 `ls -l`
- 11:46 `linux file system`
- 12:52 `path`

Simple Commands in Linux - 1

link

- 01:10 `cd..`
- 01:28 `cd enter`
- 01:35 `cd -`
- 02:12 `cd ~`
- 03:24 `date -R`
- 04:16 `cal`
- 05:06 `ncal`
- 05:40 `free`
- 05:56 `free -h`
- 07:04 `groups`
- 07:48 `ls -l output anatomy`
- 10:00 `file types`
- 12:26 `ls -i name (inode)`

- 13:24 `permissions`
- 17:17 `chmod g-w file/folder`
- 18:29 `chmod o-x file/folder`
- 19:35 `chmod 700 file/folder`
- 21:25 `cp file file_new`
- 21:54 `mv file ..`
- 22:43 `mv command to rename file`
- 25:49 `alias`
- 27:32 `hardlinks (inode)`
- 38:29 `text vs binary file`
- 38:47 `file filename`

Simple Commands in Linux - 2

link

- 00:35 Multiple `/` in path
- 02:25 `ls -l dir`
- 02:48 `ls -ld`
- 03:25 Multiple options in one hyphen
- 05:00 Long options
- 07:20 `less`
- 07:50 `cat`
- 08:45 `more`
- 09:15 `head`
- 09:30 `head -n 5`
- 09:50 `tail`
- 10:10 `tail -n 5`
- 10:31 `wc file`
- 11:30 `wc -l file`
- 11:40 `which`
- 12:00 `whatis`
- 13:00 Less is more
- 14:10 `apropos`
- 16:50 `man -k`
- 17:05 `help`
- 17:30 `info`
- 18:16 `type`
- 19:00 `file`
- 19:51 `alias`
- 21:14 `unalias`
- 23:10 `touch`
- 23:40 `mkdir`
- 23:52 `cp`
- 25:02 `rm`
- 25:10 `rmdir`

- 25:52 `rm -r`
- 26:02 `rm -i`
- 27:30 Automatic Recursion in `mv` but not in `cp`
- 28:00 `cp -r`
- 29:00 Links
- 30:16 `ln -s (softlink)`
- 30:59 `ln (hardlink)`
- 32:42 `stat`
- 33:10 `du`
- 34:35 In memory file system `/proc` and `/sys`
- 36:00 `/proc/cpuinfo`
- 37:00 `/proc/version`
- 37:15 `uname -a`
- 37:30 `/proc/meminfo`
- 37:45 `free`
- 38:00 `/proc/partitions`
- 38:14 `df`
- 40:00 Process ids
- 40:30 `/sys`
- 40:45 `/sys/bus/usb/devices`

Week 2

Command line editors - Part 01

link

- 01:00 `Different editors`
- 06:25 `ed`
- 11:30 `man ed & info ed`
- 12:00 `q to quit`
- 12:30 `P for prompt`
- 13:00 `1 to go to line 1`
- 13:02 `$ to go to last line`
- 13:13 `,p to print entire buffer`
- 13:23 `2,3p to print lines 2 and 3`
- 13:55 `/hello/ to search for hello`
- 14:10 `+ to go to next line, - to go to previous line`
- 14:40 `;p to print from current line to end`
- 14:45 `%p to print entire buffer`
- 15:00 `. to print current line`
- 15:15 `! to execute shell commands`
- 15:45 `r !date to insert output of date command in current line`
- 16:00 `w to write to file`
- 16:10 `q to quit`
- 16:45 `.d to delete current line`
- 17:45 `a to append after current line (press . to end)`
- 18:55 `s/a/b/ to substitute a with b in current line`
- 21:05 `f to show filename`
- 22:05 `5,6j to join lines 5 and 6`
- 22:40 `m1 to move current line after line 1`
- 23:40 `u to undo`
- 25:00 `%s/\(.\*\)/PREFIX \1/ to add PREFIX to all lines`
- 27:35 `Summary`

Command line editors - Part 02

link

- 01:02 `readlink -f dir (find the final softlink)`
- 03:00 `nano demo`
- 05:55 `nano summary`
- 08:08 `vi`
- 10:30 `movement in vi`
- 12:50 `command mode`
- 18:03 `move cursor to specific positions`
- 18:39 `i (INSERT mode)`
- 19:00 `esc (Command mode)`
- 19:14 `a (INSERT mode)`

- 19:39 `dd (delete line)`
- 19:45 `p (paste deleted line at a different location)`
- 20:08 `:wq (write, save and quit)`
- 20:53 `dw (delete the current word)`
- 21:20 `2dw (delete two words)`
- 21:32 `x (delete one char)`
- 22:32 `press space in command mode to move right char by char`
- 22:43 `:f (show current file name)`
- 23:04 `:se nu / :se nonu (show/hide line numbers)`
- 23:28 `yyyy/p (copy/paste lines)`
- 24:58 `:1,5s/line/LINE/ (search and replace FIRST occurrence of word 'line' in every line)`
- 25:36 `:1,5s/line/LINE/g (search and replace ALL occurrences of word 'line' in every line)`
- 26:26 `:%s/hello/hola/g (search and replace in ENTIRE BUFFER)`

Command line editors - Part 03

link

- 00:31 `scp (Transfer file b/w systems)`
- 01:14 `untar (like unzip)`
- 02:52 `DOS format handling`
- 03:14 `vi -b file (open file in binary mode)`
- 03:27 `:%s/^M//g (search and remove ^M (carriage return) in ENTIRE BUFFER)`
- 04:57 `ctrl-f ctrl-d ctrl-u`
- 05:23 `:n (go to line number n)`
- 05:48 `r (replace char)`
- 06:22 `/word ENTER n (find and scroll through all occurrences of the word in file)`
- 07:04 `:%/word/new_word/g (replace word with new_word in entire file)`
- 08:02 `A (append at the end of the line)`
- 08:28 `R (replace until ESC is pressed)`
- 09:16 `cw (replace a word)`
- 11:00 `repeat a command many times using a number`
- 16:50 `emacs summary`

Networking Commands and SSH

link

- 0:26: Overview of accessing remote machines in a private network and connecting to the internet.
- 4:49: Protecting private networks through hierarchical addressing and routing rules.

- 9:14: Understanding standard port numbers and their mapping to services for remote computer access.
- 13:55: Enhancing server security with multiple layers of protection and Security Enhanced Linux mode.
- 18:19: Essential networking commands for checking machine status, network activities, and public access.
- 22:55: Explanation of multiple IP addresses on a remote machine due to virtual machines and containers, along with private network setup.
- 27:58: Understanding IP address aliases and reverse lookup in networking
- 32:43: Connecting to multiple machines remotely using SSH and accessing an Ubuntu Linux machine for free.
- 37:34: Demonstration of checking if SELinux is enabled, accessing a web server, and observing file system protection.

Week 3

Linux process management

link

- 00:45 `sleep`
- 02:00 `coproc`
- 03:58 `kill`
- 04:23 run process in background using `&`
- 04:50 `fg (bring to foreground)`
- 05:00 `ctrl-c (kill foreground process)`
- 05:16 `two ways of killing`
- 05:43 `jobs`
- 07:44 `top`
- 08:59 `ctrl-z (suspend process)`
- 11:29 `echo $-`
- 12:29 `child shell`
- 15:37 `History`
- 16:00 `!n`
- 16:30 `!!`
- 17:37 `Brace Expansion`
- 19:56 `Multiple Commands on a single line`
- 22:29 `exit codes`
- 26:22 `kill process running in separate shell`
- 27:00 `ps -e`
- 28:11 `exit code for child processes`
- 30:07 `bc (bench calculator)`
- 30:27 `ctrl-d (quit or exit)`
- 30:57 `Why learn Exit Codes?`

Combining commands and files

link

- 00:40 `Ways of combination ; , && , ||`
- 02:54 `Use of () - Runs commands in a subshell`
- 03:00 `$BASH_SUBSHELL`
- 04:18 `Subshells within Subshells`
- 06:22 `&& and || demo`
- 10:08 `File Descriptors`, stdin 0, stdout 1, stderr 2
- 11:57 `command > file`
- 16:31 `hwinfo`
- 19:11 `cat > file`
- 22:31 `cat`
- 24:14 `command >> file`
- 27:48 `command1 >> file; command2 >> file; command3 >> file`
- 28:56 `cat >> file`

Redirections

link

- 00:14 `command 2> file`
- 03:11 `command > file1 2> file2`
- 07:52 `command < file`
- 09:44 `command > file1 2>&1 (redirect output and error both to the same file i.e. file1)`
- 13:54 `pipe | operator`
- 17:17 `command1 | command > file`
- 19:10 `/dev/null`
- 22:29 `command | tee file`
- 26:00 `diff`
- 26:22 `command1 | tee file1 file2 | command2`
- 28:02 `command1 >2 /dev/null | tee file1 file2 | command2`

Software Management - Part 01

link

- 04:27 `Check type of operating system`
- 06:41 `Check type of kernel and architecture`
- 09:48 `apt`
- 10:19 `apt-cache search pkg`
- 11:37 `apt-cache pkgnames (see all packages installed on the system)`
- 12:28 `apt-cache pkgnames nm (all packages starting with nm)`
- 12:53 `apt-cache show nmap (show details of pkg nmap)`
- 21:23 `checksums`

Software Management - Part 02

link

- 01:40 `Accessing sudoers file`
- 03:40 `/var/log`
- 05:40 `/etc/apt`
- 05:51 `cat sources.list`
- 07:30 `cd sources.list.d`
- 09:15 `update`
- 10:49 `upgrade`
- 13:26 `auto-remove`
- 14:24 `remove package`
- 15:10 `install package`
- 15:38 `reinstall package`
- 18:29 `var/lib/dpkg`
- 23:00 `dpkg -l pattern`

- 23:35 `dpkg -L package`
- 24:11 `dpkg -s package`
- 24:48 `dpkg -S pattern`
- 28:03 `dpkg-query -W -f='${Section} ${binary:Package}\n'`
- 28:13 `dpkg-query -W -f='${Section} ${binary:Package}\n' | less`
- 28:40 `dpkg-query -W -f='${Section} ${binary:Package}\n' | sort | less`
- 30:42 `dpkg-query -W -f='${Section} ${binary:Package}\n' | grep pattern`
- 31:56 `Installing a deb package`

Week 4

Pattern Matching - Part 01

link

- 01:30 `Regex`
- 03:42 `Why Regex?`
- 05:00 `Special Characters in Regex`
- 09:15 `Character Classes`
- 10:36 `Back references`
- 12:00 `Operator Precedence`
- 14:03 `grep pattern file`
- 15:41 `cat file | grep pattern`
- 16:25 `cat file | grep 'pattern.pattern'`
- 17:16 `cat file | grep 'pattern$'`
- 18:02 `cat file | grep '\.'`
- 19:10 `cat file | grep '^pattern'`
- 20:21 `cat file | grep 'pattern\b'`
- 21:42 `cat file | grep 'patt[ern]'`
- 23:04 `cat file | grep 'pat.\*tern'`
- 23:50 `cat file | grep '\bpat.\*tern'`
- 25:31 `cat file | grep 'pat[1-5]tern'`
- 27:37 `cat file | grep 'pat[^1-5]tern'`
- 28:00 `cat file | grep 'pattern\{2,4\}'`
- 29:58 `cat file | grep '\(pattern\)'`
- 30:20 `cat file | grep '\(pattern\).\*\1'`
- 32:41 `cat file | grep '\(pattern\)\{2, 3\}'`
- 34:00 `cat file | egrep 'M+'`
- 34:32 `cat file | egrep '^M+'`
- 35:00 `cat file | egrep '^M\*'`
- 35:25 `cat file | egrep 'M*a' vs 'M.*a'`
- 37:00 `cat file | egrep '(ma)+'`
- 37:25 `cat file | egrep '(ma)\*'`
- 37:58 `cat file | egrep '(ED|ME)'`

Pattern Matching - Part 02

link

- 01:12 `dpkg-query | grep`
- 08:03 `cat file | grep '[[:alpha:]]'`
- 08:30 `cat file | grep '[[:alnum:]]'`
- 10:51 `cat file | grep '[[:digit:]]'`
- 11:32 `cat file | grep '[[:cntrl:]]'`
- 12:07 `cat file | grep -v '[[:cntrl:]]'`
- 12:26 `cat file | grep '[[:punct:]]'`
- 13:15 `cat file | grep '[[:lower:]]'`

- 14:04 `cat file | grep '[[:upper:]]'`
- 14:57 `cat file | grep '[[:print:]]'`
- 16:17 `cat file | grep '[[:blank:]]'`
- 16:57 `cat file | grep '[[:space:]]'`
- 17:30 `cat file | grep '[[:graph:]]'`
- 18:27 `skip all empty lines`
- 21:40 `egrep '[[:digit:]]{12}' file`
- 22:30 `egrep '\b[[:digit:]]{6}\b' file`
- 24:16 `egrep '\b[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]][[:digit:]]{2}\b' file (Matching Roll Numbers)`
- 25:32 `urls`
- 28:43 `cut -c 1-4 file`
- 29:32 `cut -c -4 file`
- 30:00 `cat file | cut -d " " -f 1`
- 33:01 `cat file | cut -d ";" -f 2 | cut "," -f 1`
- 34:26 `grep version of above command`
- 35:30 `cat file | cut -d "/" -f 3 | cut -d " " -f 1 | head -n 19 | tail -n 1`

Week 5

$hell variables

link

- 02:45 Frequently used Shell variables ($USER, $HOME, $PATH, $PWD, $HOSTNAME)
- 03:30 Special Shell variables ($0, $1 to $9, $#, $-, $@, $? and $$)
- 04:30 `$$` (PID of the current shell)
- 04:45 `$?` (Exit status of the last command), 0 for success, non-zero for failure
- 06:30 Shell flags and `$-` to list them.
- 07:30 `echo` command
- 08:00 `echo` with multiple arguments
- 08:30 `echo` with quotes
- 10:46 `echo 'hello "` - Multi line echo and nesting quotes
- 12:45 `echo $USER`
- 13:00 Variables not expanded in single quotes
- 15:30 Escaping `$` using `\$`
- 17:14 `printenv`
- 17:44 `env`
- 17:52 `set`
- 19:04 `date`
- 19:20 `date -R`
- 20:48 How to run unalised commands using `\date` or full path `/usr/bin/date`
- 21:50 `$PATH` variable
- 22:20 `special shell variables`
- 23:22 `ps`
- 24:15 `ps --forest`
- 24:35 `ps -ef`
- 25:21 `ps -f`
- 26:36 `ps -e`

Shell Variables - Part 1

link

- 00:34 variable basic rules
- 01:21 Exporting variable
- 01:44 Using variable
- 02:18 Remove variable
- 02:49 Test if variable is set or not.
- 03:36 print default value of variable if set or display a substitute message 33:15 and 36:15
- 04:16 set default value of variable if not already set 34:19
- 04:40 reset variable if already set 37:13

- 05:06 List of variables
- 05:39 Length of string
- 05:55 slice of string
- 06:35 matching pattern
- 07:20 keep matching pattern
- 07:41 replace matching pattern
- 08:13 replace matching pattern by location
- 08:34 changing case
- 09:10 restricting value types
- 09:41 remove restrictions
- 10:20 Indexed arrays
- 12:26 Associative arrays (like dictionary in python)
- 22:24 use of{ } with variables
- 24:38 variable availability to shells and subshells (export)
- 27:21 modifying exported variable in child shell
- 28:40 setting command output to variable
- 36:36 Show error message if variable not set without substituting (`${var?msg}`)

Shell Variables - Part 2

link

- 00:41 `echo ${!H*}` - List of shell variables starting H
- 02:43 `echo ${#USER}` length of a variable
- 04:33 `echo ${USER:2:2}` slice of string stored in a variable
- 06:00 `echo ${USER: -3:2}` negative index from right
- 07:43 date command options
- 10:58 pattern matching using # and ## (delete from start)
- 13:28 pattern matching using % and %% (delete from end)
- 14:51 mixing ## and %%
- 16:22 replacing substring using / or //
- 18:11 replacing substring using /# or /%
- 24:57 changing case using , and „
- 28:49 restricting variable types using declare
- 30:09 declare -i var (int)
- 31:11 declare -l var (lower)
- 32:41 declare -u var (upper)
- 33:44 declare +u var
- 34:45 declare -r var (read only)
- 36:37 declare -a arr (Indexed Arrays)
- 38:28 echo `${#arr[@]}` (Number of elements in an array)
- 38:42 echo `${arr[@]}` (Elements of an array)
- 39:00 echo `${!arr[@]}` (Indices of an array)
- 40:46 unset ‘arr[index]’ (delete the element of an array present at given index)

- 41:31 arr+=(element) (append an element to an array)
- 42:33 populate array in one go
- 42:29 declare -A dict (Associative array - like dictionaries in python)
- 46:44 pass output of a command to an array

Week 6

Some Command line Utilities

link

- 00:30 `find`
- 00:45 `tar` and `gzip`
- 00:50 `make`
- 01:30 `find` options
- 01:45 `-name`, `-type`, `-atime`, `-ctime`, `-regex`, `-exec`, `-print`
- 03:30 File packaging and compression
- 04:00 `tar` to package files and folders
- 04:30 `gzip` to compress files
- 05:15 `zip` to compress files
- 05:30 `compress`, `gzip`, `bzip2`, `xz`, `7z`
- 07:00 Time taken to compress vs compression ratio
- 08:00 `make` utility
- 09:30 `find` example
- 10:00 `find $HOME -print | wc -l` to count files in home directory
- 10:30 `find . -m -2` to find files modified in last 2 days
- 11:15 `find . -m +30` to find files modified more than 30 days ago
- 12:00 `find /usr -type d -name "man?" -print` to find man pages folders
- 13:30 `find . -size +10M -exec ls -lsh {} \;` to find files larger than 10M
- 16:00 `find . -name '*.jpg' -exec ls -sh {} \;` to find all jpg files
- 18:15 `du -sh` to find disk usage of folders
- 18:40 `tar -cvf logfiles.tar logfiles/` create a tar file
- 19:30 `gzip logfiles.tar` to compress the tar file
- 20:30 `bzip2 logfiles.tar` to compress the tar file smaller but slower
- 21:30 `compress logfiles.tar` to compress the tar faster but larger
- 22:00 `gunzip` or `gzip -d` and `bzip2 -d` to decompress
- 23:00 `tar -xvf logfiles.tar` to extract the tar file
- 24:00 `make` utility to backup files

Overview of Shell Scripts

link

- 00:00 Scripts
- 02:00 types of scripts
- 04:00 shebang
- 04:50 sourcing vs executing
- 06:42 script location
- 07:35 bash environment
- 09:04 `echo "hello world"`
- 09:30 `printf`

- 10:14 read from command line using `read`
- 10:40 arguments `$0`, `$1`, `$2`, `$#`, `$@`, `$*`
- 12:50 command substitution
- 13:25 for loop
- 14:00 `IFS` (Internal Field Separator)
- 14:35 case (switch case) end with esac
- 15:24 if loop , end with fi
- 15:58 if loop conditions
- 19:10 comparisons
- 19:27 file comparisons
- 20:00 while loop
- 20:50 functions

Bash Scripts - Part 01

link

- 00:44 `vi s1.sh`
- 01:52 `. s1.sh` or `source s1.sh` (Run script in same shell)
- 02:48 `echo $$` (print PID)
- 03:50 `./s1.sh` (execute script using path)
- 04:00 `chmod +x s1.sh` (make script executable)
- 04:54 different PID based on if the script is executed using path vs using source
- 05:40 `ps --forest` (inside script)
- 07:08 availability of shell variables to parent shell based on execution method (path vs source)
- 08:43 detecting how the script is invoked inside the script `echo $0`
- 09:49 arguments
- 12:58 if statement
- 16:36 for loop
- 21:56 grep within for loop

Week 7

Bash Scripts - Part 2A

link

- 01:00 Debugging using `set -x` and `set +x` or `bash -x ./script.sh`
- 01:15 Combining conditions using `&&` and `||` outside `[ ]` or inside `[[ ]]`
- 03:15 Arithmetic in shell using `(( ))`, `let`, `expr`, `$(( ))` and `bc`
- 06:00 Operators inside `(( ))` and `let`
- 08:00 `str : regex` and `str =~ regex` in `[[ ]]`
- 08:00 `match str regex`
- 08:15 `substr str start length`
- 08:30 `index str char` give position of char in str (first occurrence)
- 08:45 `length str`
- 09:35 Regex to match only digits in a line `^[0-9]+$`
- 14:30 `bc` - bench calculator
- 18:00 Regex `[oO]ctav[aeiou]*` to match Octave, octave, Octav, Octavio, etc
- 18:30 Match using `expr $str : $regex`
- 22:20 heredoc
- 24:00 ignore tabs in heredoc using `<<-`
- 27:30 `IFS=:` to change delimiter

Bash Scripts - Part 2B

link

- 00:30 `if`, `if-else`, `if-elif-else`
- 03:15 `case`
- 07:45 `for((i=0;i<10;i++))`
- 12:00 Redirecting loops to files
- 15:15 `time <command>` to measure time taken
- 16:00 `break` out of loop
- 17:30 `break` from outer loop using `break 2`
- 20:15 `continue` to skip rest of the loop
- 23:00 `shift` to shift arguments to left by 1 or `[n]` if provided
- 26:30 `exec` to replace current shell with another command

Bash Scripts - Part 2C

link

- 00:30 `eval` to evaluate a command
- 07:00 `getopts` to parse command line options
- 11:30 `select` loop

Week 8

Automating Scripts

link

- 00:30 `cron` and `at` commands
- 02:00 Job definition in `cron`
- 05:00 Startup scripts
- 10:00 `crontab -e` to edit cron jobs

Stream editor sed

link

- 11:02 `sed -e "" file`
- 11:50 `sed -n -e "" file`
- 12:48 `sed -e "=" file`
- 13:28 print a particular line
- 14:15 importance of -n option
- 15:38 ‘$p'vs'$!p’ vs “$p”
- 16:17 address range
- 16:37 combine commands
- 17:39 print every n th line
- 18:52 regex address
- 19:36 `/regex/,+n`
- 20:01 delete a particular line
- 21:17 delete a range of lines
- 21:33 `/regex/d`
- 21:49 search and replace
- 24:32 extended regex
- 26:58 range-end as regex
- 30:09 regex to regex
- 32:02 insert header and footer
- 33:49 insert or append at any line
- 34:16 insert or append @ a regex address
- 36:18 change a line
- 38:06 sed script file
- 43:47 join lines (demonstrates how to read one more line)
- 47:25 Debug

Week 9

AWK Programming Part 1

link

- 00:30 `awk` command (Aho, Weinberger, and Kernighan)
- 02:20 Execution Model - Read, Process, Write
- 02:30 Each line is a record (`\n` separated)
- 02:40 Each record is a set of fields (space or tab separated)
- 05:15 Running awk in command line `cat file | awk '{print $1}'`
- 07:00 Blocks in awk (BEGIN, END, pattern-action)
- 08:30 `awk` -> `gawk` (GNU awk) (use `realpath` to find the path of awk)
- 09:00 `awk -f script file` with shebang `#!/usr/bin/awk -f`
- 11:30 Multiple BEGIN and END blocks (order matters)
- 12:50 `$0` for entire record, `$1` for first field, `$NF` for last field
- 14:10 Built in Variables of AWK

Built in Variables of AWK

- `ARGC` - The number of command-line arguments passed to the awk script.

  Example: `echo "hello world" | awk 'END{print ARGC}'` returns 1.

- `ARGV` - An array that contains the command-line arguments passed to the awk script.

  Example: `awk 'BEGIN{for(i in ARGV) print ARGV[i]}' file1 file2` prints the values of ARGV array for file1 and file2.

- `ENVIRON` - An array that contains the values of environment variables.

  Example: `awk 'BEGIN{print ENVIRON["HOME"]}'` prints the value of the HOME environment variable.

- `FILENAME` - The name of the current input file being processed.

  Example: `awk '{print FILENAME}' file1 file2` prints the name of the current file being processed.

- `FNR` - The current record number in the current input file.

  Example: `awk '{print FNR, $0}' file1 file2` prints the line number and contents of each line in both files.

- `FS` - The field separator used by awk to separate fields in a record.

  Example: `awk 'BEGIN{FS=","}{print $1}' file1` prints the first field of each record in file1, assuming that the fields are separated by commas.

- `NF` - The number of fields in the current record.

  Example: `awk '{print NF}' file1` prints the number of fields in each record in file1.

- `NR` - The current record number (across all input files).

  Example: `awk '{print NR, $0}' file1 file2` prints the line number and contents of each line in both files, counting lines across both files.

- `OFMT` - The output format for numbers.

  Example: `awk 'BEGIN{OFMT="%.3f"}{print $1/3}' file1` prints the first field of each record in file1, divided by 3 and rounded to 3 decimal places.

- `OFS` - The output field separator used by awk.

  Example: `awk 'BEGIN{OFS=","}{print $1, $2}' file1` prints the first and second fields of each record in file1, separated by commas.

- `ORS` - The output record separator used by awk.

  Example: `awk 'BEGIN{ORS="\n\n"}{print $0}' file1` prints the contents of file1, with an extra blank line between each record.

- `RS` - The input record separator used by awk.

  Example: `awk 'BEGIN{RS=","}{print $0}' file1` prints all characters in file1, separated by commas.

- `RLENGTH` - The length of the string matched by the match function.

  Example: `awk 'BEGIN{print match("hello world", /world/)}'` prints the length of the string matched by the regular expression /world/ in the string "hello world".

- `RSTART` - The starting position of the string matched by the match function.

  Example: `awk 'BEGIN{match("hello world", /world/); print RSTART}'` prints the starting position of the string matched by the regular expression /world/ in the string "hello world".

- `SUBSEP` - The separator used to separate multiple subscripts in an array.

  Example: `awk 'BEGIN{a["hello","world"]=1; print a["hello", "world"]}'`

- 16:33 Pattern matching in awk
- 18:05 Types of blocks
- 19:20 Operators in awk
- 20:00 Ternary, Array Membership, Regex
- 21:15 Built in functions in awk
- 24:45 Regex in action block
- 27:15 Match certain field with regex
- 29:30 Comparison operators in action blocks
- 30:00 `FS="[ .;:-]"` to set multiple FS using regex

AWK Programming Part 2

link

- 00:15 Arrays in awk (Associative, sparse, index may not be integers)
- 00:30 `arr[index]=value`
- 00:45 `for (var in arr)`
- 01:00 `delete arr[index]`
- 01:05 Types of loops in awk (for, while, do-while, C-style for, if, if-else, if-else-if, switch)
- 02:00 Payroll Management System using awk script on text file
- 06:15 User defined functions in awk (`awk -f lib.awk -f script.awk`)
- 11:00 Print Formatting using `printf` in awk
- 12:00 Use awk as a programming language to generate random numbers
- 15:30 How to comment in awk script using `#`
- 17:00 Processing file with million lines using awk
- 19:30 Spreadsheet applications cannot process such big files, but awk can
- 21:45 Process web server log book using awk
- 24:45 Get the first field of each line in a file using awk
- 25:50 `substr` function in awk
- 28:30 `date --date="5 days ago" +%d/%m/%Y` to get date 5 days ago
- 29:30 `sprintf` function in awk to format strings and store in a variable
- 29:40 `cmd | getline var` to read output of a command into a variable
- 30:00 `match` function in awk to match a regex in a string
- 34:00 How to sort a file using `sort` using `-n` for numeric sort and `-r` for reverse sort
- 34:30 `dig` to get IP address of a domain
- 35:00 `dig -x` to get domain name from IP address
- 35:30 `dig +noall +answer -x` to get one line answer

Week 10

Version Control - Part 01

link

- 00:22: Introduction to managing code versions and collaboration in programming projects.
- 05:19: Overview of Version Control Systems
- 10:26: Importance of preventing hardware failures in storage systems.
- 14:58: Data storage technology overview: RAID systems improve speed and safety through disk mirroring.
- 20:01: Illustration of RAID configuration with data distributed across multiple hard disks for fault tolerance.
- 24:49: Understanding the Git protocol for remote synchronization and version control.
- 29:47: Security measures for account verification using phone numbers to prevent identity theft.
- 34:28: Introduction to using git and sharing screens for guidance on navigating options.
- 39:34: Managing access tokens, creating repositories, and working with personal access tokens for GitHub.
- 44:18: Introduction to setting up a remote server and working with Git accounts.
- 49:26: Importance of Personal Access Token for authentication in version control process.

Version Control - Part 02

link

- 0:59: Setting up GitHub account and two-factor authentication for version control.
- 5:42: Managing branches and merging changes in version control.
- 9:51: Introduction to setting up GitHub for version control in a tutorial session.
- 15:15: Setting up a new repository and avoiding special characters in naming conventions.
- 20:05: Cloning a private repository requires authentication and password input.
- 26:54: Setting up Git configuration for a new repository.
- 31:50: Troubleshooting network issues, checking status, and preparing to push changes in Git.
- 38:57: Introduction to Branching and Merging in Version Control System
- 44:17: Merging branches in Git using command prompt to combine different versions.

Knowing your Hardware

link

- 01:24 `hwinfo`
- 03:03 `lshw`
- 03:49 `lshw -c display`
- 04:10 CPU info (`cat /proc/cpuinfo`)
- 05:41 partitions (`cat /proc/partitions`)
- 06:11 `lsblk`
- 07:27 `lspci`
- 08:18 `free`
- 08:56 DIMM modules (`sudo dmidecode --type memory`)
- 11:42 `hardinfo`
- 12:37 `clinfo` (OpenCL details)
- 14:12 `upower`
- 17:12 hard disk statistics (`sudo hdparm -Tt /dev/sda`)
- 18:20 `df -h`
- 20:22 `iostat -dx /dev/sdb`
- 21:31 `ifconfig` (now replaced by `ip`)

Prompt String

link

- 00:40 Types of shells
- 01:55 Bash prompts (PS1, PS2, PS3, PS4)
- 02:55 Escape Sequences in Prompt
- 04:25 Python interactive mode prompts (ps1, ps2 in sys)
- 04:55 `$PS1`
- 06:55 `\t` for time and `\d` for date
- 07:35 `\#` to list command history number
- 08:15 `source ~/.bashrc` to reset prompt
- 09:00 `$PS2` for unclosed brackets or quotes
- 11:00 `$PS3` for select loop prompt
- 12:00 `$PS4` for trace with `set -x`
- 13:30 Python prompts `sys.ps1` and `sys.ps2`

Managing Storage

link

- 0:49: Understanding logical volume management in Linux for efficient disk space allocation.
- 2:51: Data protection and storage optimization through RAID controllers and modes.
- 5:17: Importance of distributed parity in storage systems and benefits of RAID 6 for data protection.

- 7:58: Overview of RAID storage configurations and their benefits in managing storage.
- 10:47: Utilizing multiple hard disks to create a single, large storage volume for efficient access and management.