G/S/A

. - any char.

\ - escape

^ - beginning of line

$ - end of line

\< - begin. of word

\> - end of word

* - expr, 0 or more

+ - 1 or more

? - 0 or 1 times

{m,n} - at least m, at most n

.* - any seq.

[,] - space or comma

GREP -E

'-v' - lines that don't match the expr

'-i' - case insensitive

'-q' - quiet (0 or 1)

SED - E

"s/r/repl./flags" - search and replace

flags: 'g' everywhere on line

'i' case insens.

"y/char/repl" - translit.

len(char) = len(repl)

"regex/d" - delete

AWK (-F for delimiter)

no "F" → 'space' delimiter

NR → current line of input

NF → nr. of fields on the line

$0 → the input line

-f prog.awk for awk file

BEGIN & END

test - for comparisons

read - read from input

find 'dir' - lists all files in a dir

sort + uniq - for ~~storing~~ sorting

-c for count

file size in 'ls -l'

expr: a = `expr ...`

for ...; do

done

if ...; then

elif ...; then

else

while: ...; do

done

#!/bin/bash

zombie proc. - proc. whose parent hasn't called 'wait()' on them; the OS keeps the child proc. in the proc. table until P asks for the exit code; 'wait' releases the zombie

fork() ret. val. → child PID for parent, 0 for child

signals - mechanisms that interrupt a proc. and make it run a certain handler

SIGKILL - cannot be redirected - ignores handlers

signal - doesn't signal anything actually

exec - name of the command + exact content of the com. ('p' - searches in the PATH)

NULL marks the end of the args. (execl, execlp, execv, execvp)

pipe → comm. channel, buffer in the mem.; close unused ends ASAP; 0 read, 1 write

→ does not allow comm. unless you inherit the pipe; descendants only

read → empty pipe - waits for data or no ~~reader~~ writer connected

write → full pipe - waits for space or no reader connected

FIFO → file on disk; create, open, delete explicitly mkfifo; rem/unlink

→ open - waits for the FIFO to be opened for the complem. op. (open in the same order)

popen - shell command from C; pclose; dup(oldfd) - creates a copy of the handle @ index oldfd; dup2(oldfd, newfd) copy of oldfd @ newfd

Process-states - what happens with a process during its lifetime

HOLD - becomes a proc. but is yet to execute; held back by the job planning and res. allocator;

READY - loaded up in mem. doesn't have procesor

RUN - is given processor by the OS, can be sent back to 'READY' to let other processes RUN

WAIT - process does i/o doesn't occupy CPU

SWAP - the mem. of a proc. is put into swap (disk region - extension of the mem.) until enough mem.

HOLD
↓
RDY
↓
W ← RUN → SW
↓
FINISH

DEADLOCK → stop a process or roll back to a prev. state (i.p.)
a) mutex exclusion
b) lock & hold while waiting
c) non-preemption
d) circular wait

SOL - always choose an order for the resources & lock them in the same order (any order)

PROC. SCHEDULING
- f.c.f.s. (small jobs might wait)
- shortest j. first (risk of starvation) need to know jobs
- priorities
- deadline scheduling

Round Robin - give jobs time quantas

REPLACING MET.

- NRU - mark every page w/ 2 bits (R, W)

	R	W	
0 -	0	0	- not R not W
1 -	0	1	
2 -	1	0	
3 -	1	1	

set bits to 1 when an op. is performed; periodically reset bits to 0

MEM. ALLOC

a) Real
   i) single user/task OS
   ii) multi - // -

fixed part.
1. absolute
2. relocatable
3. variable partitions (mem. fragm. - sol. coalescing)

b) Virtual
   i) Paged
   ii) Segmented
   iii) Paged - Segmented

LOADING MET.

- load all pages into RAM
- first page them when needed
- locality principle
  ↳ pre-fetch neighbouring pages

- LRU - n x m matrix for n pages
  page accessed - fill its line w/ 1 and col. w/ 0
  Choose a page w/ the minimum line sum

cache of N pages, placing physical page k

direct cache org. : k → k % N - leads to cache trashing
set-cache org. : first free cache slot - slow, but no trashing
set-associative cache: group cache pages, find the group of a page using '%', then first empty

hard links - created only by the root ; h.l. goes to the data
- multiple i-nodes pointing to the same data → h.l.

symbolic links - s.l. go to the files
- multiple files pointing to the same i-node → s.l.