Memory

Date: 22 – 26 February 2021 **Week:** 3

Task I: Tutorial questions

1. What is RowHammer? What are the potential solutions?
2. Comparing the three levels of cache.
3. Explain the differences between RAID 1, 5 and 10.
4. Describe the concept of processing in memory.

Task II: Practical exercises

1. Linux commands.

A process is a program in a state of execution. A process can run in a foreground, background or be suspended. Normally the shell does not return the Linux prompt until the process has finished running.

Using the `ps -ef` command, list all the processes that are running.

1.1 Running processes in background and foreground

To send a process to background add & at the end of the command. To run sleep in the background, type:

```
$ sleep 10 &
```

The & runs the job in the background and returns the prompt straight away, allowing you to run other programs while waiting for that one to finish. The system returns the job number and the PID of the process. At the prompt, type :

```
$ sleep 1000
```

This command will take a while to complete. You can suspend the process running in the foreground by typing ^Z, i.e., hold down the [Ctrl] key and press[z]. Then to put it in the background, type :

```
$ bg
```

When a process is running, backgrounded or suspended, it will be entered onto a list along with a job number. To examine this list, type :

```
$ jobs
```

To restart (foreground) a suspended process, use fg %jobnumber.

For example, to restart sleep 1000, type

```
$ fg %1
```

Typing fg with no job number foregrounds the last suspended process.

1.2 Killing a process

Sometimes a process has to be killed because is not responding or it is in an indefinite loop. To kill a job running in the foreground, press (Control - c). For example, type:

```
$ sleep 100
```

```
[Ctrl + c]
```

To kill a suspended or background process, type

```
$ kill %jobnumber
```

For example, run :

```
$ sleep 100 &
```

```
$ jobs
```

If it is job number 4, type

```
$ kill %4
```

To check whether this has worked, examine the job list again to see if the process has been removed.

1.3 ps (process status)

Alternatively, processes can be killed by finding their process numbers (PIDs) and using kill PID_number

```
$ sleep 1000 &
```

```
$ ps
```

PID TT S TIME COMMAND 20077 pts/5 S 0:05 sleep 1000

To kill off the process sleep 1000, type

```
$ kill 20077
```

and then type ps again to see if it has been removed from the list. If a process refuses to be killed, uses the -9 option, i.e. type

```
$ kill -9 20077
```

Note: It is not possible to kill off other users' processes !

2. Vi exercise (Refer to the vi cheat-sheet at the end of this document).
   Start **vi** without a file name.

```
$ vi
```

   Enter the text: "The quick fox jumps over the lazy dog"
   - Find a way to save this text into a file "viFile.txt" without quitting.
   - Find a way to quit the editor now.
   - List the content of this file with the more and the cat commands.

   Open the "viFile.txt" file again using the vi command.

   Find a way to insert the word "brown" in between "quick" and "fox".

   Save and quit vi.

   Using the wget command below, download the book The Adventure of Sherlock Holmes

```
$ wget https://gutenberg.org/files/1661/1661-0.txt
```

   Open this book in vi. The file name is "1661-0.txt".

   Search for the occurrence of the word **Holmes.** Note the line number.

   Close the book.

   Using the head command, list the first 20 lines from the book

```
$ head -20 1661-0.txt
```

   Research a command to count the total number of words in the book.

   Research a command to count how many lines in the book.

Document your answers in the learning journal, including screenshots of the Linux commands

vi Editor “Cheat Sheet”

Invoking vi:	vi *filename*	
Format of vi commands:	*[count][command]*	(count repeats the effect of the command)

Command mode versus input mode

Vi starts in command mode. The positioning commands operate only while vi is in command mode. You switch vi to input mode by entering any one of several vi input commands. (See next section.) Once in input mode, any character you type is taken to be text and is added to the file. You cannot execute any commands until you exit input mode. To exit input mode, press the escape (**Esc**) key.

Input commands (end with Esc)

a	Append after cursor
i	Insert before cursor
o	Open line below
O	Open line above
:r *file*	Insert *file* after current line

Any of these commands leaves vi in input mode until you press **Esc**. Pressing the **RETURN** key will not take you out of input mode.

Change commands (Input mode)

cw	Change word (Esc)
cc	Change line (Esc) - blanks line
c$	Change to end of line
r*c*	Replace character with *c*
R	Replace (Esc) - typeover
s	Substitute (Esc) - 1 char with string
S	Substitute (Esc) - Rest of line with text
.	Repeat last change

Changes during insert mode

<ctrl>h	Back one character
<ctrl>w	Back one word
<ctrl>u	Back to beginning of insert

File management commands

:w *name*	Write edit buffer to file *name*
:wq	Write to file and quit
:q!	Quit without saving changes
ZZ	Same as :wq
:sh	Execute shell commands (<ctrl>d)

Window motions

<ctrl>d	Scroll down (half a screen)
<ctrl>u	Scroll up (half a screen)
<ctrl>f	Page forward
<ctrl>b	Page backward
/string	Search forward
?string	Search backward
<ctrl>l	Redraw screen
<ctrl>g	Display current line number and file information
n	Repeat search
N	Repeat search reverse
G	Go to last line
*n*G	Go to line *n*
:*n*	Go to line *n*
z<CR>	Reposition window: cursor at top
z.	Reposition window: cursor in middle
z-	Reposition window: cursor at bottom

Cursor motions

H	Upper left corner (home)
M	Middle line
L	Lower left corner
h	Back a character
j	Down a line
k	Up a line
^	Beginning of line
$	End of line
l	Forward a character
w	One word forward
b	Back one word
f*c*	Find *c*
;	Repeat find (find next *c*)

Deletion commands

dd or *n*dd	Delete *n* lines to general buffer
dw	Delete word to general buffer
d*n*w	Delete *n* words
d)	Delete to end of sentence
db	Delete previous word
D	Delete to end of line
x	Delete character

Recovering deletions

p	Put general buffer after cursor
P	Put general buffer before cursor

Undo commands

u	Undo last change
U	Undo all changes on line

Rearrangement commands

yy or Y	Yank (copy) line to general buffer
“*z*6yy	Yank 6 lines to buffer *z*
yw	Yank word to general buffer
“*a*9dd	Delete 9 lines to buffer *a*
“*A*9dd	Delete 9 lines; Append to buffer *a*
“*a*p	Put text from buffer *a* after cursor
p	Put general buffer after cursor
P	Put general buffer before cursor
J	Join lines

Parameters

:set list	Show invisible characters
:set nolist	Don’t show invisible characters
:set number	Show line numbers
:set nonumber	Don’t show line numbers
:set autoindent	Indent after carriage return
:set noautoindent	Turn off autoindent
:set showmatch	Show matching sets of parentheses as they are typed
:set noshowmatch	Turn off showmatch
:set showmode	Display mode on last line of screen
:set noshowmode	Turn off showmode
:set all	Show values of all possible parameters

Move text from file *old* to file *new*

vi *old*	
“*a*10yy	yank 10 lines to buffer *a*
:w	write work buffer
:e *new*	edit new file
“*a*p	put text from *a* after cursor
:30,60w *new*	Write lines 30 to 60 in file *new*

Regular expressions (search strings)

^	Matches beginning of line
$	Matches end of line
.	Matches any single character
*	Matches any previous character
.*	Matches any character

Search and replace commands

Syntax:

```
:[address]s/old_text/new_text/
```

Address components:

.	Current line
n	Line number n
.+m	Current line plus m lines
$	Last line
/string/	A line that contains "string"
%	Entire file
[addr1],[addr2]	Specifies a range

Examples:

The following example replaces only the **first** occurrence of Banana with Kumquat in each of 11 lines starting with the current line (.) and continuing for the 10 that follow (.+10).

```
:.,.+10s/Banana/Kumquat
```

The following example replaces **every** occurrence (caused by the `g` at the end of the command) of `apple` with `pear.`

```
:%s/apple/pear/g
```

The following example removes the last character from every line in the file. Use it if every line in the file ends with `^M` as the result of a file transfer. Execute it when the cursor is on the first line of the file.

```
:%s/.$//
```