

OS-LAB-1

Members: 2017-EE-118
2017-EE-119
2017-EE-135

1. In this question, we will understand the hardware configuration of your working machine using the /proc filesystem.

- (a) Run command more /proc/cpuinfo and explain the following terms: processor and cores. [Hint: Use lscpu to verify your definitions.]
- (b) How many cores does your machine have?
- (c) How many processors does your machine have?
- (d) What is the frequency of each processor?
- (e) How much physical memory does your system have ?
- (f) How much of this memory is free ?
- (g) What is total number of number of forks since the boot in the system ?
- (h) How many context switches has the system performed since bootup ?

SOLUTION)

- A) Run command more /proc/cpuinfo and explain the following terms: processor and cores.
[Hint: Use lscpu to verify your definitions.]

CPU is an electronic circuit inside the computer that carries out instruction to perform arithmetic, logical, control and input/output operations

The **core** is an execution unit inside the **CPU** that receives and executes instructions.

```
khamad@khamad-S551L8:~$ more /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 69
model name     : Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz
stepping       : 1
microcode      : 0x26
cpu MHz        : 952.290
cache size     : 4096 KB
physical id    : 0
siblings       : 4
core id        : 0
cpu cores      : 2
apicid         : 0
initial apicid : 0
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
```

- B) How many cores does your machine have?
Core(s) per socket: 2
- C) How many processors does your machine have?
CPU(s): 4
- D) What is the frequency of each processor?
cpu MHz : 1322.489

E) How much physical memory does your system have ?

```
khamad@khamad-S551LB:~$ more /proc/meminfo
MemTotal:      12150052 kB
MemFree:       9814600 kB
MemAvailable:  10518432 kB
Buffers:       82628 kB
Cached:        1013128 kB
SwapCached:    0 kB
Active:        1297572 kB
Inactive:      630048 kB
```

Using command `more /proc/meminfo`

MemTotal: 12150052 kB

MemFree: 8373276 kB

MemAvailable: 9709424 kB

Cached: 1664156 kB

G) What is the total number of forks since the boot in the system ?

```
khamad@khamad-S551LB:~$ vmstat -f
3139 forks
```

Using command `vmstat -f`

33138 forks

H) How many context switches has the system performed since bootup ?

```
khamad@khamad-S551LB:~$ pid=307
khamad@khamad-S551LB:~$ grep ctxt /proc/$pid/status
voluntary_ctxt_switches:      88
nonvoluntary_ctxt_switches:   6
```

Using command `grep ctxt /proc/$pid/status`

voluntary_ctxt_switches: 88

nonvoluntary_ctxt_switches: 0

2. In this question, we will understand how to monitor the status of a running process using the top command. Compile the program cpu.c given to you and execute it in the bash or any other shell of your choice as follows.

```
$ gcc cpu.c -o cpu
$ ./cpu
```

This program runs in an infinite loop without terminating. Now open another terminal, run the

top command and answer the following questions about the cpu process.

- (a) What is the PID of the process running the cpu command?
- (b) How much CPU and memory does this process consume?
- (c) What is the current state of the process? For example, is it running or in a blocked state or a zombie state?

Solution)

```
khamad@khamad-S551LB: ~
top - 19:53:55 up 13 min, 1 user, load average: 0.84, 0.83, 0.75
Tasks: 231 total, 2 running, 223 sleeping, 0 stopped, 6 zombie
%Cpu(s): 25.4 us, 0.9 sy, 0.0 ni, 71.5 id, 0.2 wa, 0.0 hi, 2.0 si, 0.0 st
MiB Mem : 11865.3 total, 9478.3 free, 1177.8 used, 1209.2 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 10213.6 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 3531 khamad    20   0   2356    584   520  R 100.0   0.0   0:36.31  cpu
    99 root      20   0     0     0     0  I   3.3   0.0   0:00.60  kworker/u8:1-phy0
 2294 root      20   0     0     0     0  I   1.7   0.0   0:00.23  kworker/u8:0-cfg80211
 1402 khamad    20   0 4558476 228400 91732  S   1.3   1.9   0:31.87  gnome-shell
 1090 khamad    20   0 1150168  91688 57552  S   0.7   0.8   0:19.35  Xorg
   266 root      19  -1 132452  51752 50096  S   0.3   0.4   0:00.64  systemd-journal
 2266 khamad    20   0 3064568 313748 165696  S   0.3   2.6   0:23.19  firefox
```

- A) What is the PID of the process running the cpu command?
PID= 32477
- B) How much CPU and memory does this process consume?
%CPU=100.0 %MEM= 0.0
- C) What is the current state of the process? For example, is it running or in a blocked state or a zombie state?
RUNNING STATE AS INDICATED IN s STATES

3. In this question, we will understand how the Linux shell (e.g., the bash shell) runs user commands

by spawning new child processes to execute the various commands.

(a) Compile the program `cpu-print.c` given to you and execute it in the bash or any other shell of your choice as follows.

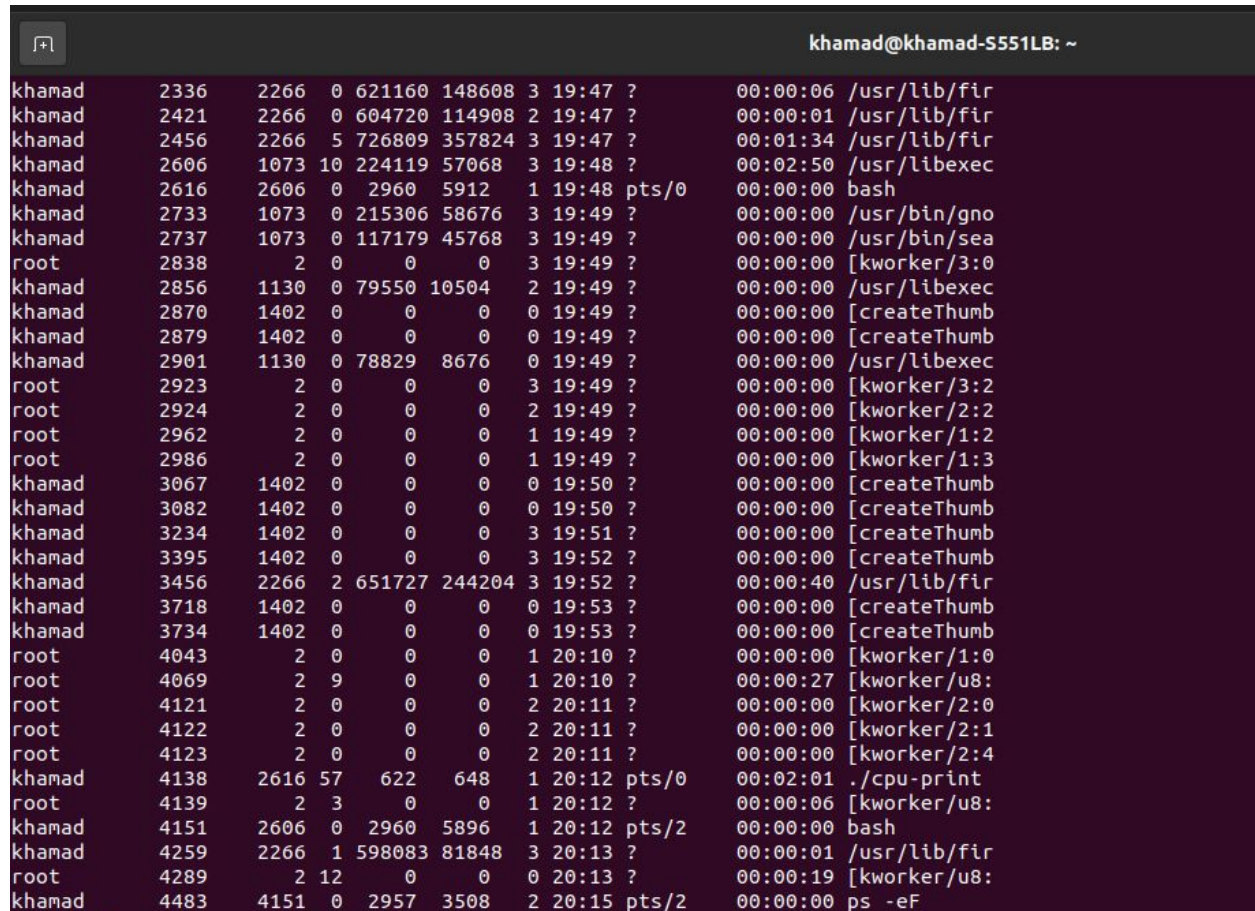
```
$ gcc cpu-print.c -o cpu-print
```

```
$ ./cpu-print
```

This program runs in an infinite loop printing output to the screen. Now, open another terminal and use the `ps` command with suitable options to find out the pid of the process spawned by the shell to run the `cpu-print` executable. You may want to explore the `ps` command thoroughly to understand the various output fields it shows.

(b) Find the PID of the parent of the `cpu-print` process, i.e., the shell process. Next, find the PIDs of all the ancestors, going back at least 5 generations (or until you reach the `init` process).

4138-> 2616-> 2606-> 1073->1 (init)



khamad	2336	2266	0	621160	148608	3	19:47	?	00:00:06	/usr/lib/fir
khamad	2421	2266	0	604720	114908	2	19:47	?	00:00:01	/usr/lib/fir
khamad	2456	2266	5	726809	357824	3	19:47	?	00:01:34	/usr/lib/fir
khamad	2606	1073	10	224119	57068	3	19:48	?	00:02:50	/usr/libexec
khamad	2616	2606	0	2960	5912	1	19:48	pts/0	00:00:00	bash
khamad	2733	1073	0	215306	58676	3	19:49	?	00:00:00	/usr/bin/gno
khamad	2737	1073	0	117179	45768	3	19:49	?	00:00:00	/usr/bin/sea
root	2838	2	0	0	0	3	19:49	?	00:00:00	[kworker/3:0]
khamad	2856	1130	0	79550	10504	2	19:49	?	00:00:00	/usr/libexec
khamad	2870	1402	0	0	0	0	19:49	?	00:00:00	[createThumb
khamad	2879	1402	0	0	0	0	19:49	?	00:00:00	[createThumb
khamad	2901	1130	0	78829	8676	0	19:49	?	00:00:00	/usr/libexec
root	2923	2	0	0	0	3	19:49	?	00:00:00	[kworker/3:2]
root	2924	2	0	0	0	2	19:49	?	00:00:00	[kworker/2:2]
root	2962	2	0	0	0	1	19:49	?	00:00:00	[kworker/1:2]
root	2986	2	0	0	0	1	19:49	?	00:00:00	[kworker/1:3]
khamad	3067	1402	0	0	0	0	19:50	?	00:00:00	[createThumb
khamad	3082	1402	0	0	0	0	19:50	?	00:00:00	[createThumb
khamad	3234	1402	0	0	0	3	19:51	?	00:00:00	[createThumb
khamad	3395	1402	0	0	0	3	19:52	?	00:00:00	[createThumb
khamad	3456	2266	2	651727	244204	3	19:52	?	00:00:40	/usr/lib/fir
khamad	3718	1402	0	0	0	0	19:53	?	00:00:00	[createThumb
khamad	3734	1402	0	0	0	0	19:53	?	00:00:00	[createThumb
root	4043	2	0	0	0	1	20:10	?	00:00:00	[kworker/1:0]
root	4069	2	9	0	0	1	20:10	?	00:00:27	[kworker/u8:
root	4121	2	0	0	0	2	20:11	?	00:00:00	[kworker/2:0]
root	4122	2	0	0	0	2	20:11	?	00:00:00	[kworker/2:1]
root	4123	2	0	0	0	2	20:11	?	00:00:00	[kworker/2:4]
khamad	4138	2616	57	622	648	1	20:12	pts/0	00:02:01	./cpu-print
root	4139	2	3	0	0	1	20:12	?	00:00:06	[kworker/u8:
khamad	4151	2606	0	2960	5896	1	20:12	pts/2	00:00:00	bash
khamad	4259	2266	1	598083	81848	3	20:13	?	00:00:01	/usr/lib/fir
root	4289	2	12	0	0	0	20:13	?	00:00:19	[kworker/u8:
khamad	4483	4151	0	2957	3508	2	20:15	pts/2	00:00:00	ps -eF

(c) We will now understand how the shell performs output redirection. Run the following command.

```
$ ./cpu-print > /tmp/tmp.txt &
```

Look at the proc file system information of the newly spawned process. Pay particular attention to where its file descriptors 0, 1, and 2 (standard input, output, and error) are pointing to.

Using this information, can you describe how I/O redirection is being implemented by the shell?

```
khamad 4151 2606 0 3122 6604 2 20:12 pts/2 00:00:00 bash
root 4289 2 11 0 0 0 20:13 ? 00:01:26 [kworker/u8:5-events_unbound]
root 4493 2 0 0 0 2 20:16 ? 00:00:00 [kworker/2:0-mm_percpu_wq]
root 4495 2 0 0 0 0 20:16 ? 00:00:00 [kworker/0:0-events]
khamad 4522 2266 4 755904 422608 2 20:17 ? 00:00:22 /usr/lib/firefox/firefox -contentproc -childID 8
root 4617 2 0 0 0 3 20:19 ? 00:00:00 [kworker/3:1-events]
khamad 4635 1402 0 0 0 1 20:19 ? 00:00:00 [createThumbnail] <defunct>
root 4661 2 0 0 0 1 20:19 ? 00:00:00 [kworker/1:1-events]
root 4666 2 0 0 0 1 20:19 ? 00:00:00 [kworker/1:2-cgroup_destroy]
root 4687 2 13 0 0 2 20:19 ? 00:00:49 [kworker/u8:4-events_unbound]
root 4758 2 0 0 0 2 20:20 ? 00:00:00 [kworker/2:2-events]
khamad 4858 1402 0 0 0 3 20:22 ? 00:00:00 [createThumbnail] <defunct>
root 4979 2 0 0 0 3 20:22 ? 00:00:00 [kworker/3:3-events]
khamad 5081 2266 0 598083 83656 0 20:24 ? 00:00:00 /usr/lib/firefox/firefox -contentproc -childID 9
root 5158 2 11 0 0 2 20:25 ? 00:00:05 [kworker/u8:0-events_unbound]
khamad 5162 4151 95 622 712 1 20:25 pts/2 00:00:07 ./cpu-print
khamad 5163 4151 0 2957 3356 3 20:26 pts/2 00:00:00 ps -eF
```

(d) Next, we will understand how the shell implements pipes. Run the following command.

```
$ ./cpu-print | grep hello
```

Once again, identify the newly spawned processes, and find out where their standard in-put/output/error file descriptors are pointing to. Use this information to explain how pipes are implemented by the shell.

```
khamad@khamad-S551LB: ~
top - 20:31:58 up 51 min, 1 user, load average: 6.35, 4.88, 3.38
Tasks: 239 total, 5 running, 224 sleeping, 0 stopped, 10 zombie
%Cpu(s): 51.3 us, 14.7 sy, 0.0 ni, 10.2 id, 23.7 wa, 0.0 hi, 0.2 si, 0.0 st
MiB Mem : 11865.3 total, 145.2 free, 1569.9 used, 10150.2 buff/cache
MiB Swap: 2048.0 total, 2038.2 free, 9.8 used, 9736.3 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR  S  %CPU  %MEM   TIME+ COMMAND
 5351 khamad    20   0   2488     712    644  S   91.0   0.0   2:55.66 cpu-print
 5162 khamad    20   0   2488     712    644  D   82.0   0.0   5:05.40 cpu-print
 2606 khamad    20   0 899228 59604 40612  R   32.3   0.5  12:41.06 gnone-terminal-
 4138 khamad    20   0   2488     648    580  R   20.7   0.0   9:52.65 cpu-print
 5352 khamad    20   0   9372    2788   2580  R   14.0   0.0   0:25.35 grep
```

Q4) Consider the two programs memory1.c and memory2.c given to you. Compile and run them one after the other. Both programs allocate a large array in memory. One of them accesses the array and the other doesn't. Both programs pause before exiting to let you inspect their memory usage. You can inspect the memory used by a process with the ps command. In particular, the output will tell you what the total size of the "virtual" memory of the process is, and how much of this is actually physically resident in memory. You will learn later that the virtual memory of the process is the memory the process thinks it has, while the OS only allocates a subset of this memory physically in RAM. Compare the virtual and physical memory usage of both programs, and explain your observations. You can also inspect the code to understand your observations.

Using the command

```
$ ps -o pid,user,%mem,command ax | sort -b -k3 -r
$ ps -aux --sort -rss
```

```
Activities Terminal 22:20 15 مارجو
khamad@khamad-S551LB: ~/Downloads
khamad@khamad-S551LB:~/Downloads$ ls
drive-download-20210315T171619Z-001.zip memory1.c
khamad@khamad-S551LB:~/Downloads$ gcc memory1.c -o memory1.c
gcc: fatal error: input file 'memory1.c' is the same as output file
compilation terminated.
khamad@khamad-S551LB:~/Downloads$ gcc memory1.c -o memory1.o
khamad@khamad-S551LB:~/Downloads$ ./memory1.c
bash: ./memory1.c: Permission denied
khamad@khamad-S551LB:~/Downloads$ ./memory1
bash: ./memory1: No such file or directory
khamad@khamad-S551LB:~/Downloads$ gcc memory1.c -o memory1
khamad@khamad-S551LB:~/Downloads$ ./memory1

Program : 'memory_1'
PID : 7987
Size of int : 4
Press Enter Key to exit.
khamad@khamad-S551LB:~/Downloads$ ps -p 7987 -o pid,user,%mem,command,ppid,vsz,rss
ps -p 7987 -o pid,user,%mem,command,ppid,vsz,rss
ps -aux --sort -rss
```

```
Activities Terminal 22:23 15 مارچ
khamad@khamad-S551LB: ~/Downloads
khamad@khamad-S551LB:~/Downloads$ ./memory2
Program : 'memory_2'
-----
PID : 8192
Size of int : 4
Press Enter Key to exit.
khamad@khamad-S551LB:~/Downloads$ ./memory2
Program : 'memory_2'
-----
PID : 8211
Size of int : 4
Press Enter Key to exit.
$ ps -aux --sort -rss
$ ps -aux --sort -rss
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	TIME	COMMAND
khamad	1073	0.0	0.0	19348	6456	?	Ss 19:41 0:01	/lib/systemd/systemd --user
khamad	1149	0.0	0.0	317056	6328	?	Ssl 19:41 0:00	/usr/libexec/gvfs-afc-volume-monitor
root	835	0.0	0.0	14060	6104	?	Ss 19:41 0:00	/sbin/wpa_supplicant -u -s -O /run/wpa
khamad	1135	0.0	0.0	382052	6080	?	Sl 19:41 0:00	/usr/libexec/gvfsd-fuse /run/user/1000/
khamad	7539	0.0	0.0	11972	6076	pts/0	Ss 22:12 0:00	bash
khamad	7868	0.0	0.0	11840	5908	pts/1	Ss 22:17 0:00	bash
root	991	0.0	0.0	239808	5872	?	Ssl 19:41 0:00	/usr/sbin/gdm3
khamad	2901	0.0	0.0	315316	5788	?	Sl 19:49 0:00	/usr/libexec/gvfsd-dnssd --spawn
khamad	1448	0.0	0.0	162868	5648	?	Sl 19:41 0:01	/usr/libexec/at-spi2-registryd --use-gn
khamad	1130	0.0	0.0	240128	5280	?	Ssl 19:41 0:00	/usr/libexec/gvfsd
khamad	6805	0.0	0.0	360052	5280	?	Sl 21:59 0:00	/usr/lib/speech-dispatcher-modules/sd
khamad	1537	0.0	0.0	231792	5160	?	Sl 19:42 0:00	/usr/libexec/gsd-disk-utility-notify
khamad	6812	0.0	0.0	360084	4988	?	Sl 21:59 0:00	/usr/lib/speech-dispatcher-modules/sd
khamad	8211	0.0	0.0	6272	4864	pts/0	S+ 22:23 0:00	./memory2
khamad	1084	0.0	0.0	240504	4696	?	Sll 19:41 0:00	/usr/bin/gnome-keyring-daemon --daemon
message+	809	0.0	0.0	8916	4672	?	Ss 19:41 0:02	/usr/bin/dbus-daemon --system --address
khamad	1479	0.0	0.0	156360	4408	?	Sl 19:41 0:00	/usr/libexec/dconf-service
khamad	1105	0.0	0.0	8652	4344	?	Ss 19:41 0:02	/usr/bin/dbus-daemon --session --address
syslog	828	0.0	0.0	224348	4332	?	Ssl 19:41 0:01	/usr/sbin/rsyslogd -n -iNONE
khamad	1760	0.0	0.0	162624	4148	?	Ssl 19:42 0:00	/usr/libexec/gvfsd-metadata
khamad	8213	0.0	0.0	12432	4092	pts/1	R+ 22:23 0:00	ps -aux --sort -rss
root	832	0.0	0.0	16928	3984	?	Ss 19:41 0:00	/lib/systemd/systemd-logind
khamad	1519	0.0	0.0	235836	3836	?	Ssl 19:42 0:00	/usr/libexec/gsd-screensaver-proxy
khamad	1162	0.0	0.0	236204	3712	?	Ssl 19:41 0:00	/usr/libexec/gvfs-goa-volume-monitor
khamad	1518	0.0	0.0	457428	3684	?	Ssl 19:42 0:00	/usr/libexec/gsd-rfkill
khamad	1154	0.0	0.0	238440	3644	?	Ssl 19:41 0:00	/usr/libexec/gvfs-gphoto2-volume-monit
khamad	1088	0.0	0.0	164348	3632	tty2	Ssl+ 19:41 0:00	/usr/lib/gdm3/gdm-x-session --run-scrip
khamad	1074	0.0	0.0	103568	3580	?	S 19:41 0:00	(sd-pam)