Shreyas Marwah

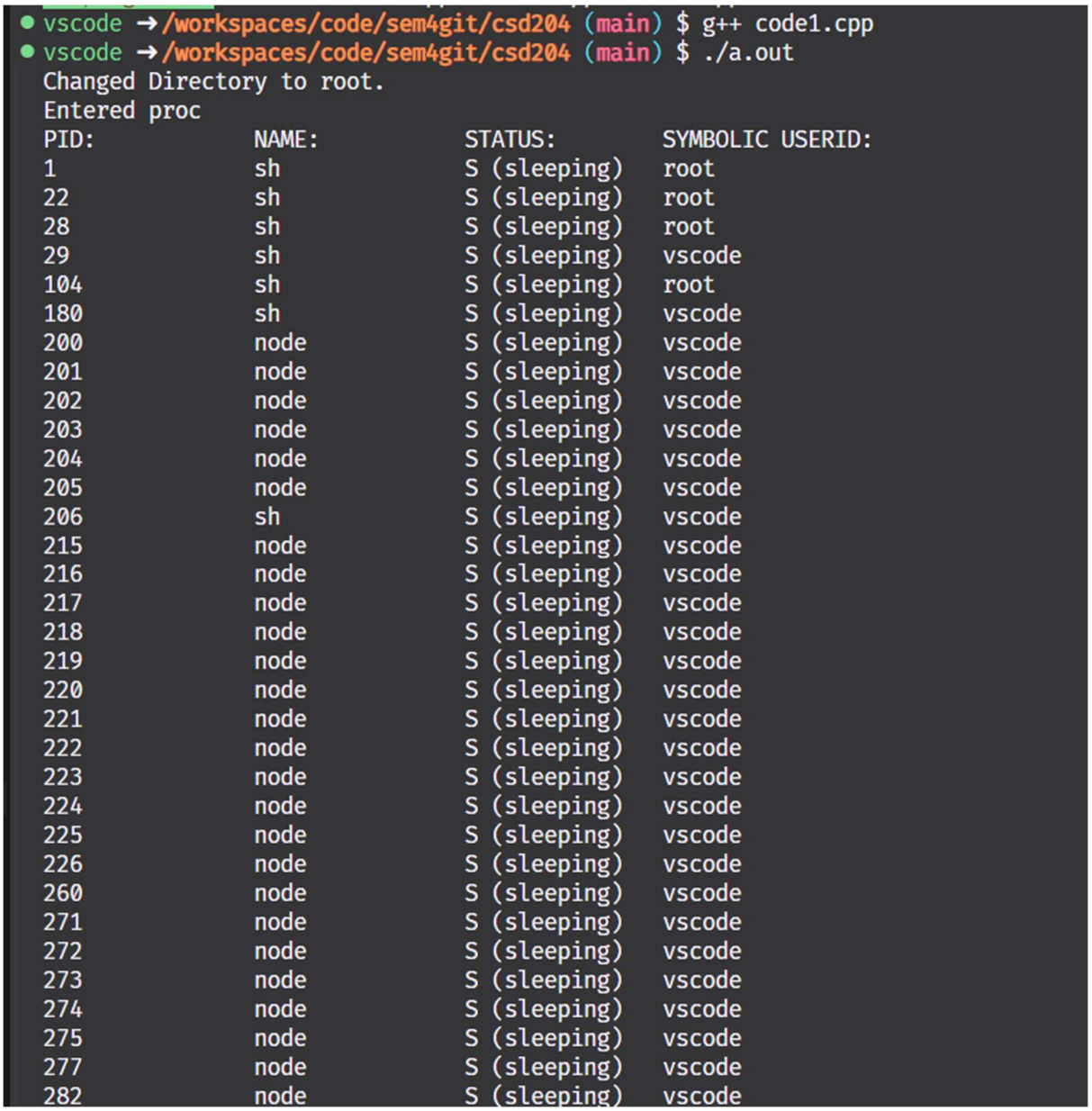
2310110604

Lab -1

Ans 1

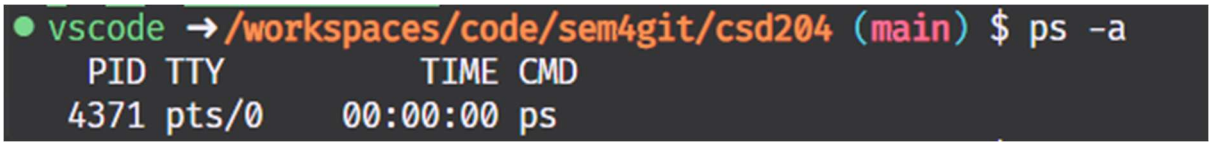
The ps -a commands lists much fewer programmes than the code that has been written in C++ (code1.cpp)

the output of code1.cpp is:



```
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ g++ code1.cpp
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ ./a.out
Changed Directory to root.
Entered proc
PID:            NAME:           STATUS:         SYMBOLIC USERID:
1               sh              S (sleeping)    root
22              sh              S (sleeping)    root
28              sh              S (sleeping)    root
29              sh              S (sleeping)    vscode
104             sh              S (sleeping)    root
180             sh              S (sleeping)    vscode
200             node            S (sleeping)    vscode
201             node            S (sleeping)    vscode
202             node            S (sleeping)    vscode
203             node            S (sleeping)    vscode
204             node            S (sleeping)    vscode
205             node            S (sleeping)    vscode
206             sh              S (sleeping)    vscode
215             node            S (sleeping)    vscode
216             node            S (sleeping)    vscode
217             node            S (sleeping)    vscode
218             node            S (sleeping)    vscode
219             node            S (sleeping)    vscode
220             node            S (sleeping)    vscode
221             node            S (sleeping)    vscode
222             node            S (sleeping)    vscode
223             node            S (sleeping)    vscode
224             node            S (sleeping)    vscode
225             node            S (sleeping)    vscode
226             node            S (sleeping)    vscode
260             node            S (sleeping)    vscode
271             node            S (sleeping)    vscode
272             node            S (sleeping)    vscode
273             node            S (sleeping)    vscode
274             node            S (sleeping)    vscode
275             node            S (sleeping)    vscode
277             node            S (sleeping)    vscode
282             node            S (sleeping)    vscode
```

the output of ps -a is:



```
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ ps -a
  PID TTY          TIME CMD
 4371 pts/0    00:00:00 ps
```

Ans 2

Code has been written in code2.cpp, the readme is code2.readme
the output is as given below

```
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ g++ code2.cpp
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ ./a.out
System Configuration:
OS Version: "Debian GNU/Linux 11 (bullseye)"
CPU Model: AMD Ryzen 5 5600H with Radeon Graphics
Total Threads: 12
Total Cores: 6
Main RAM: 7617 MB
```

Ans 3

a) In the output of more /proc/cpuinfo command, it seems that the words ‘thread’ and ‘processor’ has been used interchangeably.
However colloquially we refer to the entire chip as a CPU which is composed of multiple cores and each core runs one or two threads.

definition of core: A core is a physical HARDWARE processing unit inside the CPU that can run code independently.
definition of processor: When hyperthreading is enabled, each core can run multiple threads simultaneously. Thus the number of processors seems to virtually increase. A logical processor is a processing unit that the os sees and schedules tasks on

the output is as given below:

```
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ more /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 25
model           : 80
model name      : AMD Ryzen 5 5600H with Radeon Graphics
stepping        : 0
microcode       : 0xffffffff
cpu MHz         : 3293.732
cache size      : 512 KB
physical id     : 0
siblings        : 12
core id         : 0
cpu cores       : 6
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtsc
 lm constant_tsc rep_good nopl tsc_reliable nonstop_tsc cpuid extd_apicid pni pclmulqdq ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hyp
rvisor lahf_lm cmp_legacy svm cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw topoext perfctr_core ssbd ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi
 erms invpcid rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves clzero xsaveerptr arat npt nrip_save tsc_scale vmcb_clean flushbyasid decod
assists pausefilter pfthreshold v_vmsave_vmload umip vaes vpclmulqdq rdpid fsrm
bugs            : sysret_ss_attrs null_seg spectre_v1 spectre_v2 spec_store_bypass srso
bogomips        : 6587.46
TLB size        : 2560 4K pages
clflush size    : 64
cache_alignment : 64
address sizes   : 48 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : AuthenticAMD
```

b) 6 cores

c) My machine has 12 processors
command used: lscpu

d) 3.3 GHz
command used: lscpu

```
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ lscpu
Architecture:                         x86_64
CPU op-mode(s):                       32-bit, 64-bit
Byte Order:                           Little Endian
Address sizes:                        48 bits physical, 48 bits virtual
CPU(s):                               12
On-line CPU(s) list:                  0-11
Thread(s) per core:                   2
Core(s) per socket:                   6
Socket(s):                            1
Vendor ID:                            AuthenticAMD
CPU family:                           25
Model:                                80
Model name:                           AMD Ryzen 5 5600H with Radeon Graphics
Stepping:                             0
CPU MHz:                              3293.732
BogoMIPS:                             6587.46
Virtualization:                       AMD-V
Hypervisor vendor:                    Microsoft
Virtualization type:                  full
L1d cache:                            192 KiB
L1i cache:                            192 KiB
L2 cache:                             3 MiB
L3 cache:                             16 MiB
Vulnerability Gather data sampling:   Not affected
Vulnerability Itlb multihit:          Not affected
Vulnerability L1tf:                   Not affected
Vulnerability Mds:                    Not affected
Vulnerability Meltdown:               Not affected
Vulnerability Mmio stale data:        Not affected
Vulnerability Reg file data sampling: Not affected
Vulnerability Retbleed:               Not affected
Vulnerability Spec rstack overflow:   Mitigation; safe RET
Vulnerability Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:             Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:             Mitigation; Retpolines; IBPB conditional; IBRS_FW; STIBP always-on; RSB filling; PBRSB-eIBRS Not affected; BHI Not affecte
                                      d
Vulnerability Srbds:                  Not affected
Vulnerability Tsx async abort:        Not affected
Flags:                                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxs
```

e) x86_64
command used: lscpu

f) Total memory: 7.4 GiB
command used: free -h

```
vscode ➜ /workspaces/code/sem4git/csd204 (main) $ free -h
               total        used        free      shared  buff/cache   available
Mem:           7.4Gi       1.2Gi       5.5Gi       3.0Mi       759Mi       5.9Gi
Swap:          2.0Gi          0B       2.0Gi
```

g) Free memory: 5.5GiB
command used: free -h

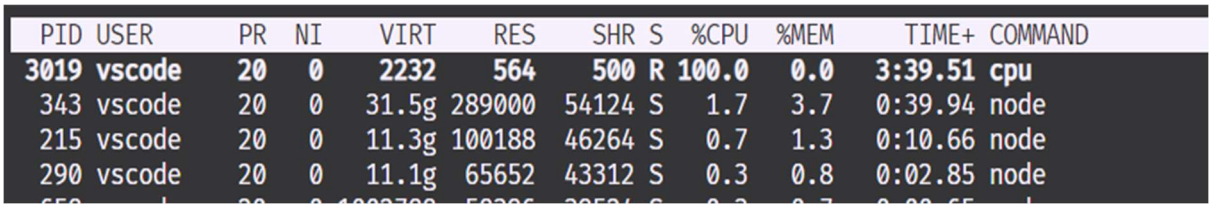
h) total number of forks: 7855
total number of context switches since the machine booted up: 9,204,638
command used for both of the above: more /proc/stat (read the stat file located in the /proc directory)

Ans 4

The four fields under ps -a are as given below:

1) PID (Process ID): A unique number assigned to each running process.
2) TTY (Terminal): The terminal associated with the process (e.g., pts/0 for an SSH session or tty1 for a physical terminal).
3) TIME (CPU Time): The total amount of CPU time the process has used.
4) CMD (Command): The name of the command that started the process.

a) Pid :3019
b) the cpu usage of cpu command is 100.0%
   the memory usage of the cpu command is 0.0%

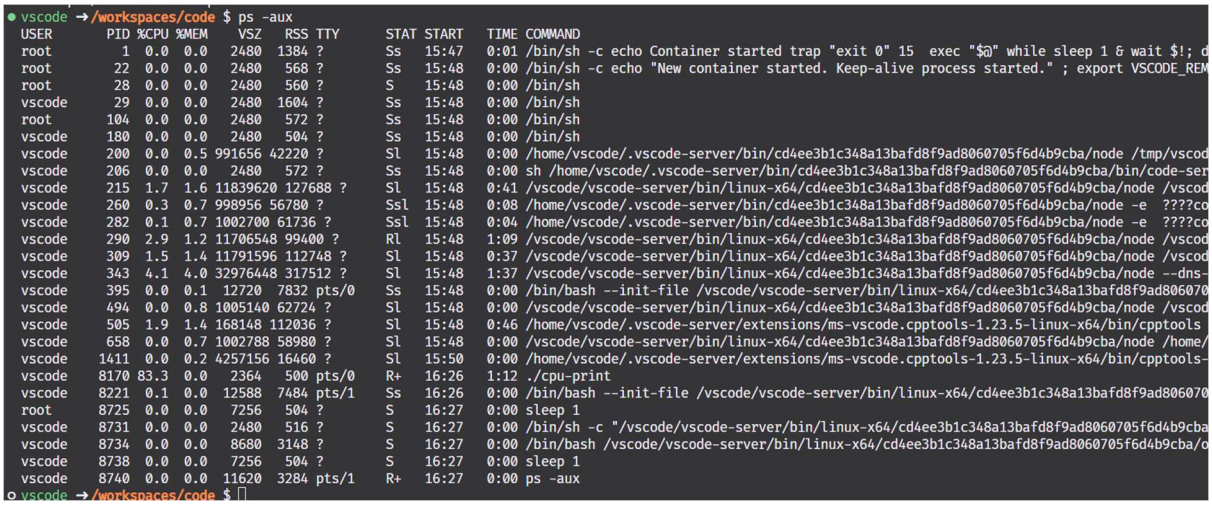


PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3019	vscode	20	0	2232	564	500	R	100.0	0.0	3:39.51	cpu
343	vscode	20	0	31.5g	289000	54124	S	1.7	3.7	0:39.94	node
215	vscode	20	0	11.3g	100188	46264	S	0.7	1.3	0:10.66	node
290	vscode	20	0	11.1g	65652	43312	S	0.3	0.8	0:02.85	node

   the command that has been used is: top
c) since the S(state) column says R(running) the cpu process in running

Ans 5

The output of ps -aux is:



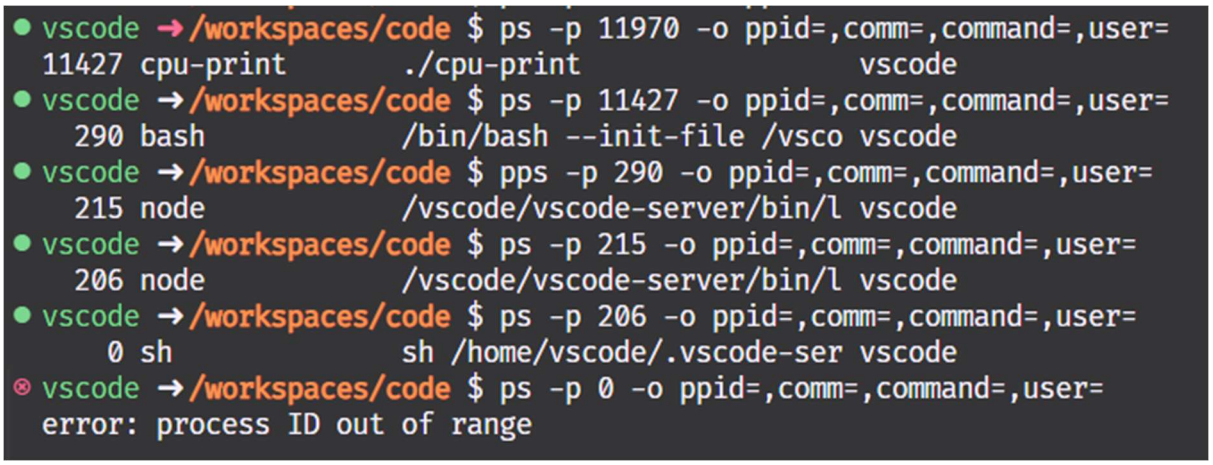
```
vscode ➜ /workspaces/code $ ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2480  1384 ?        Ss   15:47   0:01 /bin/sh -c echo Container started trap "exit 0" 15  exec "$@" while sleep 1 & wait $!; d
root          22  0.0  0.0   2480   568 ?        Ss   15:48   0:00 /bin/sh -c echo "New container started. Keep-alive process started." ; export VSCODE_REM
root          28  0.0  0.0   2480   560 ?        S    15:48   0:00 /bin/sh
vscode        29  0.0  0.0   2480  1604 ?        Ss   15:48   0:00 /bin/sh
root         104  0.0  0.0   2480   572 ?        Ss   15:48   0:00 /bin/sh
vscode       180  0.0  0.0   2480   504 ?        Ss   15:48   0:00 /bin/sh
vscode       200  0.0  0.5 991656 42220 ?        Sl   15:48   0:00 /home/vscode/.vscode-server/bin/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node /tmp/vscod
vscode       206  0.0  0.0   2480   572 ?        Ss   15:48   0:00 sh /home/vscode/.vscode-server/bin/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/bin/code-ser
vscode       215  1.7  1.6 11839620 127688 ?     Sl   15:48   0:41 /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node /vscod
vscode       260  0.3  0.7 998956 56780 ?        Ssl  15:48   0:08 /home/vscode/.vscode-server/bin/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node -e  ????co
vscode       282  0.1  0.7 1002700 61736 ?       Ssl  15:48   0:04 /home/vscode/.vscode-server/bin/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node -e  ????co
vscode       290  2.9  1.2 11706548 99400 ?      Rl   15:48   1:09 /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node /vscod
vscode       309  1.5  1.4 11791596 112748 ?     Sl   15:48   0:37 /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node /vscod
vscode       343  4.1  4.0 32976448 317512 ?     Sl   15:48   1:37 /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node --dns-
vscode       395  0.0  0.1  12720  7832 pts/0    Ss   15:48   0:00 /bin/bash --init-file /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad806070
vscode       494  0.0  0.8 1005140 62724 ?       Sl   15:48   0:00 /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node /vscod
vscode       505  1.9  1.4 168148 112036 ?       Sl   15:48   0:46 /home/vscode/.vscode-server/extensions/ms-vscode.cpptools-1.23.5-linux-x64/bin/cpptools
vscode       658  0.0  0.7 1002788 58980 ?       Sl   15:48   0:00 /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/node /home/
vscode      1411  0.0  0.2 4257156 16460 ?       Sl   15:50   0:00 /home/vscode/.vscode-server/extensions/ms-vscode.cpptools-1.23.5-linux-x64/bin/cpptools-
vscode      8170 83.3  0.0   2364   500 pts/0    R+   16:26   1:12 ./cpu-print
vscode      8221  0.1  0.0  12588  7484 pts/1    Ss   16:26   0:00 /bin/bash --init-file /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad806070
root        8725  0.0  0.0   7256   504 ?        S    16:27   0:00 sleep 1
vscode      8731  0.0  0.0   2480   516 ?        S    16:27   0:00 /bin/sh -c "/vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba
vscode      8734  0.0  0.0   8680  3148 ?        S    16:27   0:00 /bin/bash /vscode/vscode-server/bin/linux-x64/cd4ee3b1c348a13bafd8f9ad8060705f6d4b9cba/o
vscode      8738  0.0  0.0   7256   504 ?        S    16:27   0:00 sleep 1
vscode      8740  0.0  0.0  11620  3284 pts/1    R+   16:27   0:00 ps -aux
vscode ➜ /workspaces/code $
```

The various fields of the pls -aux commands are as follows:

a) User: name of the user
b) PID: process id
c) %cpu: percentage of cpu used by each process
d) %mem: percentage of physical memory used by each process
e) vsz: virtual memory size that is used by the process in kB
f) RSS: resident set size: the amount of RAM that the process is using in kB
g) tty: the id of the terminal where the process is running, if it is not running in any terminal then a '?' symbol is observed here
h) STAT: the process state (r for running, s for sleeping etc)
i) start: the time that the process was started (24 hour format)
j) time: the total cpu time that the process has used so far
k) command: the command that started the process

part B

The ppid of the process is as given below



1) cpu-print (id: 11970, parent: bash)
2) bash (id: 11427, parent: node)
3) node (id: 290, parent: node)
4) node (id: 215, parent:sh)
5) sh (id: 206) (no parent, PPID: 0)

part C

command used is ls -l /proc/<insert_pid>/fd

Shell redirection works by the shell manipulating a process's file descriptors *before* the process even starts. When you run ./cpu-print > /tmp/tmp.txt, the shell first creates a copy of itself (fork). Then, it uses dup2 to remap the standard output (file descriptor 1) of this copy to /tmp/tmp.txt. Finally, it replaces the copy's program image with cpu-print (exec). cpu-print inherits these remapped file descriptors, so when it writes to stdout, the output goes to the file, without cpu-print itself needing to know anything about redirection. The shell sets up the I/O channels before the program runs.

part D

```
● vscode ➜ .../code/sem4git/csd204/Helping-codes (main) $ gcc cpu-print.c -o cpu-print
● vscode ➜ .../code/sem4git/csd204/Helping-codes (main) $ ./cpu-print | grep hello &
  [1] 23551
```

```
● vscode ➜ .../code/sem4git/csd204/Helping-codes (main) $ ls -l /proc/23551/fd
  total 0
  lr-x------ 1 vscode vscode 64 Feb  4 17:18 0 -> 'pipe:[346371]'
  lrwx------ 1 vscode vscode 64 Feb  4 17:18 1 -> /dev/pts/0
  l-wx------ 1 vscode vscode 64 Feb  4 17:18 19 -> /home/vscode/.vscode-server/data/logs/20250204T154802/remoteagent.log
  lrwx------ 1 vscode vscode 64 Feb  4 17:18 2 -> /dev/pts/0
  l-wx------ 1 vscode vscode 64 Feb  4 17:18 20 -> /home/vscode/.vscode-server/data/logs/20250204T154802/ptyhost.log
```

part e)

cd and history are shell built-ins. cd *must* be built-in because changing directories affects the shell's internal state; an external program couldn't change the shell's own working directory. history, which manages the command history, is also a shell built-in. ls and ps are external executables. When you type ls or ps, the shell locates and executes the corresponding program (e.g., /bin/ls, /bin/ps). These programs then interact with the operating system to retrieve file listings or process information and display the results. The shell simply launches these external utilities.

Answer 6

the codes have been compiled and run:

we use pgrep memory1 and pgrep memory2 to get the pids of the running processes

```
● vscode ➜ /workspaces/code $ pgrep memory1
  24812
● vscode ➜ /workspaces/code $ pgrep memory2
  25089
● vscode ➜ /workspaces/code $ ps -p 24812,25089 -o pid,vsz,rss,comm
    PID    VSZ   RSS COMMAND
  24812   6152   564 memory1
  25089   6156  4376 memory2
```

We can see that the virtual size that the OS allocates is similar for both the programmes, but the physical allocation is different. Hence we can conclude the following:

The operating system uses a technique called *demand paging*. When a program allocates memory, the OS doesn't necessarily allocate physical RAM immediately. It just creates a mapping in the virtual address space. Physical memory is only allocated when the program actually *tries* to access a particular page of that memory. This is why the RSS of the program that doesn't access the array is much smaller—only the pages that are actively used (like the program's code and some initial data) are loaded into RAM. The rest of the large array exists only in the virtual address space until it's accessed.

By comparing the VSZ and RSS values, you can see how the OS efficiently manages memory, only allocating physical RAM as needed. This allows programs to have large virtual address spaces without requiring equally large amounts of physical RAM.

Answer 7

5000 copies have been made using the command: ./make-copies.sh

```
 foo4367.pdf  foo4742.pdf  foo618.pdf   foo994.pdf
foo1365.pdf  foo1740.pdf  foo2115.pdf  foo2491.pdf  foo2867.pdf  foo3241.pdf  foo3617.pdf  foo3993.pdf
 foo4368.pdf  foo4743.pdf  foo619.pdf   foo995.pdf
foo1366.pdf  foo1741.pdf  foo2116.pdf  foo2492.pdf  foo2868.pdf  foo3242.pdf  foo3618.pdf  foo3994.pdf
 foo4369.pdf  foo4744.pdf  foo62.pdf    foo996.pdf
foo1367.pdf  foo1742.pdf  foo2117.pdf  foo2493.pdf  foo2869.pdf  foo3243.pdf  foo3619.pdf  foo3995.pdf
 foo437.pdf   foo4745.pdf  foo620.pdf   foo997.pdf
foo1368.pdf  foo1743.pdf  foo2118.pdf  foo2494.pdf  foo287.pdf   foo3244.pdf  foo362.pdf   foo3996.pdf
 foo4370.pdf  foo4746.pdf  foo621.pdf   foo998.pdf
foo1369.pdf  foo1744.pdf  foo2119.pdf  foo2495.pdf  foo2870.pdf  foo3245.pdf  foo3620.pdf  foo3997.pdf
 foo4371.pdf  foo4747.pdf  foo622.pdf   foo999.pdf
foo137.pdf   foo1745.pdf  foo212.pdf   foo2496.pdf  foo2871.pdf  foo3246.pdf  foo3621.pdf  foo3998.pdf
 foo4372.pdf  foo4748.pdf  foo623.pdf
foo1370.pdf  foo1746.pdf  foo2120.pdf  foo2497.pdf  foo2872.pdf  foo3247.pdf  foo3622.pdf  foo3999.pdf
 foo4373.pdf  foo4749.pdf  foo624.pdf
foo1371.pdf  foo1747.pdf  foo2121.pdf  foo2498.pdf  foo2873.pdf  foo3248.pdf  foo3623.pdf  foo4.pdf
 foo4374.pdf  foo475.pdf   foo625.pdf
vscode ➜ .../sem4git/csd204/Helping-codes/disk-files (main) $ cd ..
```

while running disk:

```
vscode ➜ .../code/sem4git/csd204/Helping-codes (main) $ iostat
Linux 5.15.167.4-microsoft-standard-WSL2 (aa976a81c657)         02/04/25        _x86_64_ (12 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.79    0.00    3.76    0.04    0.00   93.41

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
loop0             0.01         0.15         0.00         0.00       1064          0          0
loop1             0.54        43.90         0.00         0.00     310259          0          0
sda               0.16        10.52         0.00         0.00      74361          0          0
sdb               0.01         0.17         0.00         0.00       1184          4          0
sdc               0.07         3.43         0.04         0.30      24213        296       2140
sdd               8.16       152.69       955.65         0.00    1079197    6754264          0
```

while running disk1:

```
● vscode ➜ .../code/sem4git/csd204/Helping-codes (main) $ iostat
  Linux 5.15.167.4-microsoft-standard-WSL2 (aa976a81c657)         02/04/25        _x86_64_        (12 CP
  U)

  avg-cpu:  %user   %nice %system %iowait  %steal   %idle
             2.76    0.00    3.76    0.04    0.00   93.44

  Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
  loop0             0.01         0.15         0.00         0.00       1064          0          0
  loop1             0.54        43.46         0.00         0.00     310259          0          0
  sda               0.16        10.42         0.00         0.00      74361          0          0
  sdb               0.01         0.17         0.00         0.00       1184          4          0
  sdc               0.07         3.39         0.04         0.30      24213        296       2140
  sdd               8.12       152.18       949.56         0.00    1086473    6779164          0
```