

CSD204 - OS - Lab01

Lalit Maurya, 2310110164

Question 1

```
~ 09:14:16 PM
ps -A
PID TTY      TIME CMD
  1 ?        00:00:00 systemd
  2 ?        00:00:00 init-systemd(Ub
  7 ?        00:00:00 init
 49 ?        00:00:00 systemd-journal
 94 ?        00:00:00 systemd-udev
105 ?        00:00:00 systemd-resolve
112 ?        00:00:00 systemd-timesyn
156 ?        00:00:00 cron
157 ?        00:00:00 dbus-daemon
164 ?        00:00:00 systemd-logind
167 ?        00:00:00 wsl-pro-service
172 hvcc0     00:00:00 agetty
179 tty1     00:00:00 agetty
192 ?        00:00:00 rsyslogd
198 ?        00:00:00 unattended-upgr
267 ?        00:00:00 SessionLeader
269 ?        00:00:00 Relay(273)
273 pts/0     00:00:00 zsh
276 pts/1     00:00:00 login
328 pts/0     00:00:00 zsh
341 pts/0     00:00:00 zsh
343 pts/0     00:00:00 zsh
344 pts/0     00:00:00 gitstatusd-linu
398 ?        00:00:00 systemd
```

```
..D204-OS/lab01 09:14:42 PM
~/devEnv/snu/CSD204-OS/lab01 main !1 72
g++ q01.cpp -o q01

~/devEnv/snu/CSD204-OS/lab01 main !1 72 09:14:46 PM
./q01
PID | Name | Status | User ID
-----|-----|-----|-----
1 | systemd | S (sleeping) | root
2 | init-systemd(Ub | S (sleeping) | root
7 | init | S (sleeping) | root
49 | systemd-journal | S (sleeping) | root
94 | systemd-udev | S (sleeping) | root
105 | systemd-resolve | S (sleeping) | systemd-resolve
112 | systemd-timesyn | S (sleeping) | systemd-timesync
156 | cron | S (sleeping) | root
157 | dbus-daemon | S (sleeping) | messagebus
164 | systemd-logind | S (sleeping) | root
167 | wsl-pro-service | S (sleeping) | root
172 | agetty | S (sleeping) | root
179 | agetty | S (sleeping) | root
192 | rsyslogd | S (sleeping) | syslog
198 | unattended-upgr | S (sleeping) | root
267 | SessionLeader | S (sleeping) | root
269 | Relay(273) | S (sleeping) | root
273 | zsh | S (sleeping) | lalit1004
276 | login | S (sleeping) | root
```

Comparison of q01.cpp against ps -A:

Similarities	Differences
<ul style="list-style-type: none">• Both iterate over all running process and display them• Both map the uid to the username using /etc/passwd	<ul style="list-style-type: none">• The program only extracts a subset of the information that ps -A does.• ps -A provides additional information such as cpu time and terminal name

Question 2

```
~ /devEnv/snu/CSD204-05/lab01 ~ main !1 ?2 09:27:45 PM
> ./q02
Kernel Version
-----
Linux version 5.15.167.4-microsoft-standard-WSL2 (root@f9c826d3017f) (gcc (GCC) 11.2.0, GNU ld (GNU Binutils) 2.37) #1 SMP Tue Nov 5 00:21:55 UTC 2024

Memory Information
-----
MemTotal:      7940460 kB
MemFree:       7270388 kB
MemAvailable:  7247664 kB

CPU Information
-----
Processor | Cores | Model Name
-----|-----|-----
0 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
1 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
2 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
3 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
4 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
5 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
6 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
7 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
8 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
9 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
10 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
11 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
12 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
13 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
14 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
15 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
16 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
17 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
18 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
19 | 10 | 12th Gen Intel(R) Core(TM) i7-12700H
```

The program parses through /proc/meminfo, /proc/version, and /proc/cpuinfo to gather information about the system. It then formats them into a table for easy visualization.

Question 3

```
more
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 154
model name     : 12th Gen Intel(R) Core(TM) i7-12700H
stepping       : 3
microcode     : 0affffffff
cpu MHz        : 2688.010
cache size     : 24576 KiB
physical id    : 0
siblings       : 20
core id        : 0
cpu cores      : 10
apicid         : 0
initial apicid: 0
fpu            : yes
fpu_exception  : yes
cpuid level    : 20
vp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
                pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc re
                p_good nopl xtopology tsc_reliable nonstop_tsc cpuid pni pclmulqdq vmx ssse3 fma cx
                16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdr
                and hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb stibp ibrs_e
                nhanced tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erm
                s invpcid rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves avx
                _vnni umip waitpkg gfni vaes vpclmulqdq rdpid movdiri movdir64b fsrm md_clear seria
                lize flush_lld arch_capabilities
vmx flags      : vnmi invpcid ept_x_only ept_ad ept_lgb tsc_offset vtrp ept vpid u
                nrestricted_guest ept_mode_based_exec tsc_scaling usr_wait_pause
bugs           : spectre_v1 spectre_v2 spec_store_bypass swapgs retbleed e1bs_gbr
sb rdfs bhii
bogomips       : 5376.02
clflush size   : 64
cache alignment: 64
address sizes  : 39 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id      : GenuineIntel
microcode     : 0affffffff
```

```
lscpu
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Address sizes:      39 bits physical, 48 bits virtual
Byte Order:         Little Endian
CPU(s):             20
On-line CPU(s) list: 0-19
Vendor ID:          GenuineIntel
Model name:         12th Gen Intel(R) Core(TM) i7-12700H
CPU family:         6
Model:              154
Thread(s) per core: 2
Core(s) per socket: 10
Socket(s):          1
Stepping:           3
BogoMIPS:           5376.02
Flags:              fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
                    cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx
                    pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology t
                    sc_reliable nonstop_tsc cpuid pni pclmulqdq vmx ssse3 fma
                    cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline
                    _timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3
                    dnowprefetch invpcid_single ssbd ibrs ibpb stibp ibrs_enh
                    anced tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust
                    bmi1 avx2 smep bmi2 erms invpcid rdseed adx smap clflush
                    opt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves avx_vnni u
                    mip waitpkg gfni vaes vpclmulqdq rdpid movdiri movdir64b
                    fsrm md_clear serialize flush_lld arch_capabilities

Virtualization features:
Virtualization:     VT-x
Hypervisor vendor:  Microsoft
Virtualization type: full
Caches (sum of all):
L1d:                480 KiB (10 instances)
L1i:                320 KiB (10 instances)
L2:                 12.5 MiB (10 instances)
L3:                 24 MiB (1 instance)
```

Part A

Processor : The term processor in `more /proc/cpuinfo` refers to the index of the processor who's information is being displayed in the current block. This is a zero indexed value that goes up to 19 on my system. Which would mean that my system has 20 processors. This is verified by `lscpu`. Note that this number is due to hyperthreading which creates the illusion of double the processors. The number of physical processors on my system is 10.

Cores : A core is an individual processing unit within a processor. A core can work on multiple threads at once via hyperthreading.

Hyperthreading allows a core to run multiple threads, 2 for my system.

Part B

By running

```
$ lscpu | grep -E 'Socket|Core|Thread|'
Model name:          12th Gen Intel(R) Core(TM) i7-12700H
Thread(s) per core:  2
Core(s) per socket:  10
Socket(s):            1
```

We can calculate number of physical cores via: $\text{numCores} * \text{numSockets} = 10 * 1 = 10$. We can account for the virtual cores via accounting for hyperthreading. We simply need to multiply the total number of physical cores with numThreads/core resulting in 20 cores total.

Part C

The number of processors is the same as the number of virtual cores on our system. Which is 20.

Part D-H

- cpu MHz : 2688.010
- Architecture: x86_64
- MemTotal: 7940460 kb ~ 7.94046 GB

- MemFree: 7239552 kb ~ 7.23955 GB
- NumForks: 1560
- Context Switches: 292997

Question 4

```

./cpu
~/devEnv/snu/CSD204-05/lab01/helping-codes main !1 72 ... 10:14:15 PM
> gcc cpu.c -o cpu

~/devEnv/snu/CSD204-05/lab01/helping-codes main !1 73 ... 10:14:23 PM
> ./cpu

```

```

top
~/devEnv/snu/CSD204-05/lab01/helping-codes main !1 72 ... 10:14:09 PM
> top
top - 22:14:48 up 56 min, 1 user, load average: 0.34, 0.08, 0.03
Tasks: 40 total, 2 running, 38 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.0 us, 0.0 sy, 0.0 ni, 95.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7754.4 total, 7055.2 free, 677.7 used, 253.6 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 7076.7 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
 1242 lalitm1+  20   0   2548   1024   936 R 100.0   0.0   0:22.83  cpu
     1 root      20   0  21796  12816  9444 S   0.0   0.2   0:00.42  systemd
     2 root      20   0   2616   1444   1320 S   0.0   0.0   0:00.00  init-syste+
     7 root      20   0   2616    132    132 S   0.0   0.0   0:00.00  init
    49 root      19  -1  66752  15940  14772 S   0.0   0.2   0:00.25  systemd-jo+
    94 root      20   0  23992   6112   4928 S   0.0   0.1   0:00.09  systemd-ud+
   105 systemd+  20   0  21452  11708   9516 S   0.0   0.1   0:00.10  systemd-re+
   112 systemd+  20   0  91020   6576   5720 S   0.0   0.1   0:00.13  systemd-ti+
   156 root      20   0   4236   2724   2504 S   0.0   0.0   0:00.00  cron
   157 message+  20   0   9660   5224   4596 S   0.0   0.1   0:00.13  dbus-daemon
   164 root      20   0   17976   8444   7424 S   0.0   0.1   0:00.07  systemd-lo+
   167 root      20   0 1756096 15840   9236 S   0.0   0.2   0:00.11  wsl-pro-se+
   172 root      20   0    3160   1076    992 S   0.0   0.0   0:00.00 agetty
   178 root      20   0    3116   1104   1016 S   0.0   0.0   0:00.00 agetty

```

Part A - C

- The process id of the ./cpu program is 1242.
- The process is occupying a 100% of CPU and 0% of MEM.
- The current state of the process is R - Running.

Question 5

Part A - B



```
~ 10:24:50 PM
> ps -A | grep cpu-print
1350 pts/2    00:02:09  cpu-print

~ 10:24:57 PM
> ps -f 1350
UID          PID    PPID  C STIME TTY          STAT       TIME CMD
lalitm1+    1350      498  64  22:21 pts/2    R+          2:17  ./cpu-print

~ 10:25:11 PM
> ps -f 498
UID          PID    PPID  C STIME TTY          STAT       TIME CMD
lalitm1+     498      497   0  21:19 pts/2    Ss          0:01  -zsh

~ 10:25:21 PM
> ps -f 497
UID          PID    PPID  C STIME TTY          STAT       TIME CMD
root         497      496   0  21:19 ?        S           0:15  /init

~ 10:25:23 PM
> ps -f 496
UID          PID    PPID  C STIME TTY          STAT       TIME CMD
root         496        2   0  21:19 ?        Ss          0:00  /init

~ 10:25:24 PM
> ps -f 2
UID          PID    PPID  C STIME TTY          STAT       TIME CMD
root          2        1   0  21:19 ?        Sl          0:00  /init

~ 10:25:26 PM
> ps -f 1
UID          PID    PPID  C STIME TTY          STAT       TIME CMD
root          1        0   0  21:19 ?        Ss          0:00  /sbin/init

~ 10:25:29 PM
> ps -f 0
error: process ID out of range
```

1. Compile and run the cpu-print program
2. Use `ps -A | grep cpu-print` to capture the relevant information.
3. Use recursive `ps -f <PPID>` until you reach PPID 0

Part C

```
..helping-codes x + v - □ x
~ /devEnv/snu/CSD204-0S/lab01/helping-codes main !1 73 10:29:24 PM
> ./cpu-print > tmp/tmp.txt &
[1] 1595

~ /devEnv/snu/CSD204-0S/lab01/helping-codes main !1 74 10:30:06 PM
> ls -l /proc/1595/fd/
total 0
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:30 0 -> /dev/pts/2
l-wx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:30 1 -> /home/lalitm1004/devEnv/snu
/CSD204-0S/lab01/helping-codes/tmp/tmp.txt
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:30 2 -> /dev/pts/2
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:30 6 -> /dev/ptmx

~ /devEnv/snu/CSD204-0S/lab01/helping-codes main !1 74 10:30:46 PM
> █
```

We can inspect the file descriptors of the `cpu-print` program by using the command `ls -l /proc/<pid>/fd/`. This shows us the following:

- File descriptor 0 (stdin): This will point to the terminal or other standard input source.
- File descriptor 1 (stdout): This will normally point to the terminal the process was executed in. But in this case, we have redirected it to the `.../tmp/tmp.txt` file.
- File descriptor 2 (stderr): This will point to the terminal and will route any errors there.

I/O redirection in the shell is achieved by manipulating the file descriptors (stdin, stdout, stderr) before the process is launched:

1. The shell closes the current standard output
2. The shell then opens the target file for writing and associates this with `fd 1`

The process does not need to know that its output is being redirected, it simply needs to write to its stdout.

Part D

```
..helping-codes x ..helping-codes x + v - □ x
~ /devEnv/snu/CSD204-0S/lab01/helping-codes main !1 74 10:42:39 PM
> ./cpu-print | grep hello &
[1] 1727 1728

~ /devEnv/snu/CSD204-0S/lab01/helping-codes main !1 74 10:43:03 PM
> ls -l /proc/1727/fd/
total 0
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 0 -> /dev/pts/4
l-wx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 1 -> 'pipe:[17939]'
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 2 -> /dev/pts/4
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 6 -> /dev/ptmx

~ /devEnv/snu/CSD204-0S/lab01/helping-codes main !1 74 10:43:17 PM
> ls -l /proc/1728/fd/
total 0
lr-x----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 0 -> 'pipe:[17939]'
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 1 -> /dev/pts/4
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 2 -> /dev/pts/4
lrwx----- 1 lalitm1004 lalitm1004 64 Feb 3 22:43 6 -> /dev/ptmx
```

The command spawns two new processes - `cpu-print` and `grep`. Like last time we can inspect their file descriptors using `ls -l /proc/<pid>/fd/`. We can see that the stdout of `cpu-print` was pointing to the stdin of `grep` via a pipe.

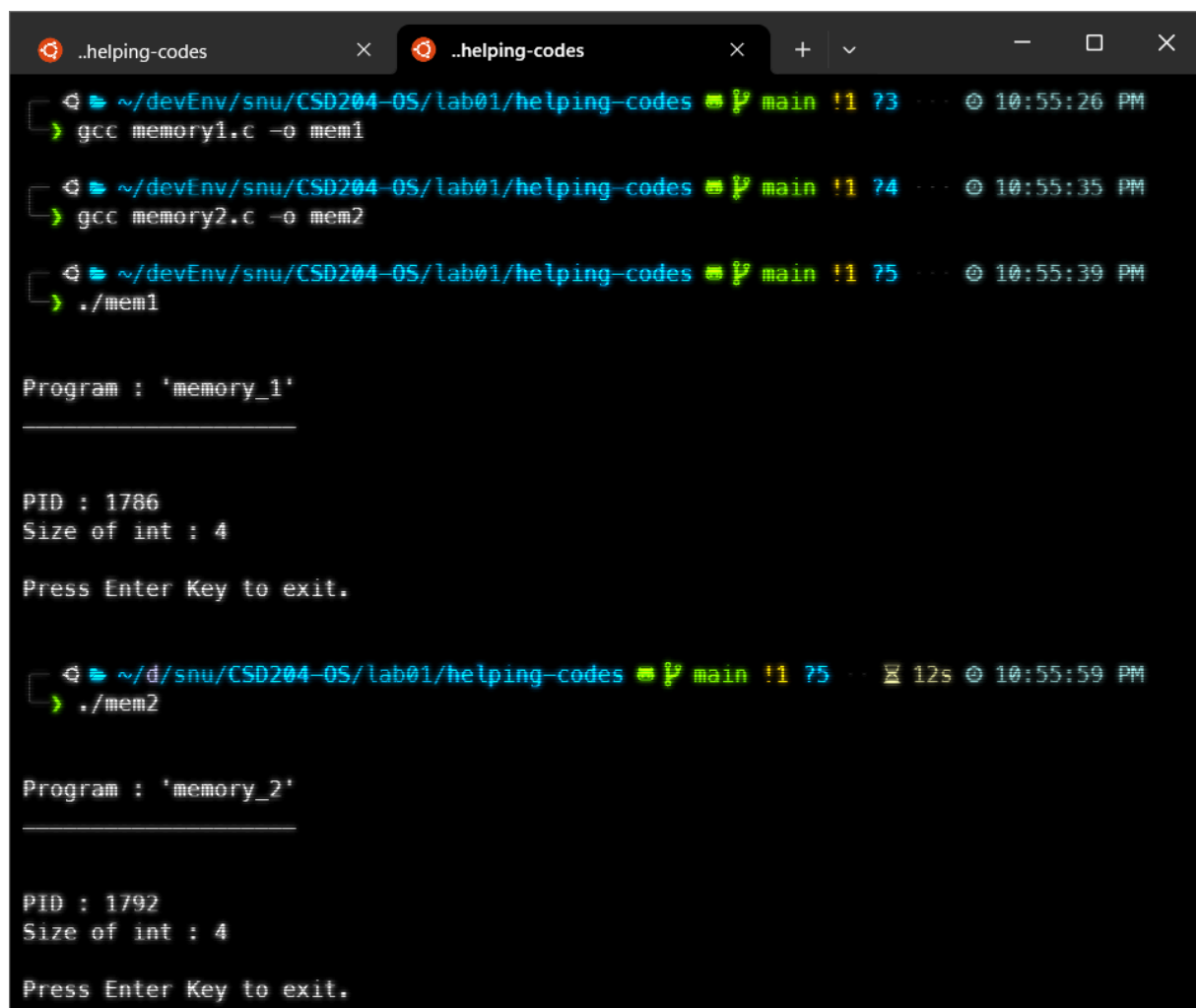
How the shell implements pipe:

1. Pipe creation: Before executing the command, the shell creates a pipe using the `pipe()` syscall. This results in a pair of file descriptors:
 - One for reading from the pipe
 - One for writing to the pipe
2. Redirection of file descriptors:
 - The stdout of `cpu-print` is redirected to the write end of the pipe.
 - The stdin of `grep` is redirected to the read end of the pipe.
3. Process execution: The shell then starts both processes.

Part E

Built In	External
<ul style="list-style-type: none">• <code>cd</code>• <code>history</code>	<ul style="list-style-type: none">• <code>ls</code>• <code>ps</code>

Question 6



```
..helping-codes x ..helping-codes x + v - □ X
~/devEnv/snu/CSD204-0S/lab01/helping-codes main !1 73 ... 10:55:26 PM
> gcc memory1.c -o mem1

~/devEnv/snu/CSD204-0S/lab01/helping-codes main !1 74 ... 10:55:35 PM
> gcc memory2.c -o mem2

~/devEnv/snu/CSD204-0S/lab01/helping-codes main !1 75 ... 10:55:39 PM
> ./mem1

Program : 'memory_1'
_____

PID : 1786
Size of int : 4

Press Enter Key to exit.

~/d/snu/CSD204-0S/lab01/helping-codes main !1 75 ... 12s 10:55:59 PM
> ./mem2

Program : 'memory_2'
_____

PID : 1792
Size of int : 4

Press Enter Key to exit.
```

```
~ x + v
10:55:17 PM
> ps u 1786
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lalitm1+  1786  0.0  0.0   6464   4856 pts/4    S+   22:55   0:00 ./mem1

10:55:53 PM
> ps u 1792
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
lalitm1+  1792  0.0  0.0   6472   5488 pts/4    S+   22:56   0:00 ./mem2
```

The commands are executed in the order:

1. ./mem1
2. ps u <pid mem1>
3. Exit mem1
4. ./mem2
5. ps u <pid mem2>
6. Exit mem2

The Virtual memory of both programs is almost the same while the Resident Set Size or the physical memory is larger in mem2. This is because of the array access done in mem2. As a result, more of the allocated memory is loaded into the physical RAM resulting in a higher RSS.

Note that I am using WSL for Windows. On a dual booted system with ubuntu installed, the RSS of mem2 is several times the RSS of mem1.

Question 7

Running disk.c

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	13.00	0.00	60.00	0.00	0	60	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	2.00	0.00	12.00	0.00	0	12	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	1401.00	130684.00	0.00	0.00	130684	0	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	2662.00	248776.00	0.00	0.00	248776	0	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	2704.00	252792.00	0.00	0.00	252792	0	0

In `disk.c`, we randomly choose to read a file from a sample of 5000. This results in practically zero rereads of a file. Which also means we see a consistent read amount when running `iostat -d 1`. This is cause we are not caching any files.

- Sustained `kB_read/s` of ~250000 kbs.

Running `disk1.c`

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	0.00	0.00	0.00	0.00	0	0	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	19.00	1044.00	32.00	0.00	1044	32	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	0.00	0.00	0.00	0.00	0	0	0

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	0.00	0.00	0.00	0.00	0	0	0
sdb	0.00	0.00	0.00	0.00	0	0	0
sdc	0.00	0.00	0.00	0.00	0	0	0

In `disk1.c`, we repeatedly read from one singular file. Which results in on initial read logged in `iostat` with all concurrent logs reporting zero reads. This is because the system caches the file which saves time on rereads.

- Initial `kB_read/s` of 1044 Kbs which then falls to 0 due to caching.