Output Screens

```
Operating System Interfaces Lab
This program explores the /proc filesystem interface


=== Process Information Lab ===
1. List all process directories
2. Read process information
3. Show system information
4. Compare file operation methods
5. Exit
Choose an option (1-5):
```

```
Listing all process directories in /proc...
Process directories in /proc:
PID      Type
---      ----
1        process
2        process
3        process
4        process
5        process
6        process
7        process
8        process
13       process
14       process
15       process
16       process
17       process
```

```
5108     process
5115     process
5228     process
5353     process
5496     process
5508     process
5545     process
5582     process
5616     process
5618     process
Found 285 process directories
SUCCESS: Process directories listed!
```

```
Choose an option (1-5): 2
Enter process ID (PID) to examine: 1

Reading information for PID 1...

--- Process Information for PID 1 ---
Name:   systemd
Umask:  0000
State:  S (sleeping)
Tgid:   1
Ngid:   0
Pid:    1
PPid:   0
TracerPid:      0
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 512
```

```
Cpus_allowed:   3
Cpus_allowed_list:      0-1
Mems_allowed:   00000000,00000001
Mems_allowed_list:      0
voluntary_ctxt_switches:        3727
nonvoluntary_ctxt_switches:     2762

--- Command Line ---
/sbin/init
SUCCESS: Process information read!
```

```
Reading system information...

--- CPU Information (first 10 lines) ---
processor       : 0
BogoMIPS        : 48.00
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm jscvt fcma lrcpc dcpop sha3 asimddp sha512 asimdfhm dit uscat ilrcpc flagm
 ssbs sb paca pacg dcpodp flagm2 frint
CPU implementer : 0x61
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0x000
CPU revision    : 0

processor       : 1

--- Memory Information (first 10 lines) ---
MemTotal:        3468516 kB
MemFree:         1351892 kB
MemAvailable:    2125152 kB
Buffers:           66200 kB
Cached:           858336 kB
SwapCached:            0 kB
Active:          1548000 kB
Inactive:         240200 kB
Active(anon):     905376 kB
Inactive(anon):        0 kB
SUCCESS: System information displayed!
```

```
Choose an option (1-5): 4

Comparing file operation methods...
Comparing file reading methods for: /proc/version

=== Method 1: Using System Calls ===
Linux version 6.14.0-37-generic (buildd@bos03-arm64-047) (aarch64-linux-gnu-gcc-14 (Ubuntu 14.2.0-19ubuntu2) 14.2.0, GNU ld (GNU Binutils for Ubuntu) 2.44) #37-Ubuntu SMP
PREEMPT_DYNAMIC Fri Nov 14 23:05:04 UTC 2025

=== Method 2: Using Library Functions ===
Linux version 6.14.0-37-generic (buildd@bos03-arm64-047) (aarch64-linux-gnu-gcc-14 (Ubuntu 14.2.0-19ubuntu2) 14.2.0, GNU ld (GNU Binutils for Ubuntu) 2.44) #37-Ubuntu SMP
PREEMPT_DYNAMIC Fri Nov 14 23:05:04 UTC 2025

NOTE: Run this program with strace to see the difference!
Example: strace -e trace=openat,read,write,close ./lab2
```

```
Choose an option (1-5): 5
Exiting program. Goodbye!
```

1)How many system calls does each method make?

1) The system calls make 4 system calls, openat once, read twice, and close.
   The library calls make 4 system calls, openat once, read twice, and close

2)What's the difference between fopen() and open()?

2)The difference is that fopen is a C library function that returns a pointer to a file structure and is buffered, and open is a low-level system call that returns an integer file descriptor and is not buffered. fopen() is useful for general file I/O, suits more applications, and is more portable. Open is used when you need low-level control over file descriptors and is usually called by fopen.

3)Why does the library method make different system calls?

3)The library method can make different system calls because they are high-level wrappers that are layered on complex operating system operations to make them simpler, portable, and optimized.

Comfortability score [0 to 5]: 3.5

Completeness score [0 to 5]: 5

A win: There were not as many bugs to fix as last time