

# LAB 6 – TASK 11

## System Calls

**John Dempsey**

COMP-232: Programming Languages  
California State University, Channel Islands

October 2, 2024

Hard Due Date: October 9, 2024

### Task 11. System Calls

System calls provide an interface to the Unix kernel. For this task, let's make sure they all work by writing a program which runs the following system calls listed below.

Remember if you need help with a system call, like `mkdir`, you can use the man pages. Since `mkdir` is a command and a system call, to view the man page for a system call you can type: `man mkdir.2` or `man -s2 mkdir`. To display the entire man page at once, you can type:

```
% man stat.2 | cat
```

You can copy the example code found in `man stat.2` for step 12 below.

To complete LAB 6, write a program called **`syscalls.c`** which:

1. Prints "System Calls".
2. Display your user name using `getpwuid(getuid())`.  
(`getpwuid` isn't actually a system call.)
3. Display your user id by using `getuid()`.
4. Display your group id by using `getgid()`.
5. Display the host you are on using `gethostname()`.
6. Display the domain name of your host using `getdomainname()`.
7. Display your current working directory using `getcwd()`.

8. Check to see if the `/etc/passwd` file exists using `access()`. Print `"/etc/passwd exists"`.
9. Check to see if you can read `/etc/passwd` file exists using `access()`.
10. Check to see if you can write `/etc/passwd` file using `access()`.
11. Using the `system()` call, run: `"id; hostname; domainname; pwd; cat /etc/passwd | grep <your_user_id>"` commands and compare the returned values with the above system calls. Replace `<your user id>` with your actual user id on the system.
12. Let's check the file status of `/etc/passwd` using `stat()`. Run **man stat.2** to view example code which you can **copy/paste** into your program. Print the following for the `/etc/passwd` file:

File status for `/etc/passwd` follows:

File type:	regular file
File l-node number:	99999
File Mode (octal):	644
File owner:	99
File group id:	99
File size in bytes:	9999
Blocks allocated:	99
Last file status change:	<date format>
Last file access:	<date format>
Last file modification:	<date format>

13. Using the `system()` call, run `"ls -l /etc/passwd"` and check above values returned.
14. Check to see if the `demo.dir` directory exists. If it exists, print `"demo.dir exists."` If `demo.dir` does not exist, create directory `demo.dir` using `mkdir` command and set mode to `0x755`.
15. Change into `demo.dir` using `chdir()`.
16. Print working directory using `getcwd()`.

17. Print your program's process id using `getpid()`.
18. Print your program's parent process id using `getppid()`.
19. Using the `system()` call, run `"ps -ef > processes.txt; cat processes.txt"`. Locate your program's process id and parent's process id in list. (You don't have to document this step, just check.)
20. Let's check your system out using `sysinfo()`. Display how long your system has been up in seconds, minutes, hours, and days; display the load averages for 1, 5, and 15 minutes; display the amount of free RAM and total RAM; and finally, display the number of processes running on your system.
21. Let's fork a process using `fork()`. The parent process will create the child.
- 22. For the child process:**
23. Print "Child Process".
24. Print "Child process id = " using `getpid()`.
25. Print "Child's parent process id = " using `getppid()`.
26. Print "-----"
27. Print output from running `"ps -ef"` using `system()`.
28. Print "-----"
- 29. For the parent process:**
30. Print "Parent Process".
31. Print "Parent process id = " using `getpid()`.
32. Print "Parent's parent process id = " using `getppid()`.

33. Using `signal()`, call `killprocess()` when a `SIGALRM` is seen. (To do this, you can add the following line of code **`signal(SIGALRM, killprocess)`**; which can be the first line in your program.)
  34. `killprocess()` is a function which displays the running processes then kills the current process. The `killprocess()` function is provided for you below (after Step 40).
  35. Next, if you are in the parent process, issue the `SIGALRM` signal in 5 seconds. If you are in the child process, issue the `SIGALRM` signal in 10 seconds.
  36. Now for both the child and parent processes go to sleep for 60 seconds using `sleep()`.
  37. Print "All Done!"
  38. Check the exit status of program using "echo \$?". Do you see 0, 10, or another number?
  39. Now, display all system calls used called by your `syscalls` program by running:  
% **`strace syscalls`**
- `strace` can be used to help locate where your code might be crashing.
40. Review the `strace` output. Look for the `write` statement for printing line "All Done!" performed in step 37.

Here is the `killprocess()` function:

```
void killprocess()
{
    int pid;

    system("ps");
    pid = getpid();
    printf("-----\n");
    printf("killprocess() was called. Kill pid = %d\n", pid);
}
```

```
printf("-----\n");  
fflush(stdout);  
kill(pid, SIGKILL);  
exit(10);  
}
```

## Expected output as an example:

```
john@oho:~$ syscalls
```

```
-----  
System Calls  
-----
```

```
Your login name is john  
Your uid is 1000  
Your group id is 1000  
/etc/passwd file entry: john:x:1000:1000:,,,:/home/john:/bin/bash  
The host name is oho  
The domain name is localdomain  
-----
```

```
Status for /etc/passwd  
-----
```

```
File status for /etc/passwd follows:
```

```
File type:          regular file  
File I-node number: 9570149208451419  
File Mode (octal): 644  
File owner:         0  
File group id:     0  
File size in bytes: 1769  
Blocks allocated:  8  
Last file status change: Tue Jan  4 20:27:17 2022  
Last file access:   Wed Jan  5 11:32:51 2022  
Last file modification: Tue Jan  4 20:27:17 2022
```

```
-rw-r--r-- 1 root root 1769 Jan  4 20:27 /etc/passwd  
-----
```

```
cd demo.dir  
-----
```

```
demo.dir exists  
CWD = /home/john/demo.dir  
-----
```

```
System Information  
-----
```

```
System load:          33984 37856 38400 (1, 5, 15 minutes)  
Uptime in seconds:   10115 seconds  
Uptime in minutes:   168 minutes  
Uptime in hours:     02 hours  
Uptime in days:      00 days  
Free memory:         7 GBs  
Total memory:        15 GBs  
#processes running: 7 processes  
-----
```

```
Processes  
-----
```

```
Process id is 1119  
Parent process id is 8  
root      1      0  0 08:44 ?          00:00:00 /init
```

```

root          7          1  0 08:44 tty1      00:00:00 /init
john          8          7  0 08:44 tty1      00:00:00 -bash
john         1119         8  0 11:32 tty1      00:00:00 a.out
john         1124        1119 0 11:32 tty1      00:00:00 sh -c ps -ef|grep 8
john         1126        1124 0 11:32 tty1      00:00:00 grep 8

```

-----  
Call fork()  
-----

Parent process

Child process

Parent process id = 1119

Child process id = 1127

Parent's parent process id = 8

Child's parent process id = 1119

In 5 seconds, the SIGALRM signal will go off in parent process.

-----  
Sleep for 60 seconds.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	08:44	?	00:00:00	/init
root	7	1	0	08:44	tty1	00:00:00	/init
john	8	7	0	08:44	tty1	00:00:00	-bash
root	152	1	0	09:41	tty2	00:00:00	/init
john	153	152	0	09:41	tty2	00:00:00	-bash
john	1119	8	0	11:32	tty1	00:00:00	a.out
john	1127	1119	0	11:32	tty1	00:00:00	a.out
john	1128	1127	0	11:32	tty1	00:00:00	sh -c ps -ef
john	1129	1128	0	11:32	tty1	00:00:00	ps -ef

-----  
In 10 seconds, the SIGALRM signal will go off in child process.

Sleep for 60 seconds.

PID	TTY	TIME	CMD
8	tty1	00:00:00	bash
1119	tty1	00:00:00	a.out
1127	tty1	00:00:00	a.out
1130	tty1	00:00:00	sh
1131	tty1	00:00:00	ps

-----  
Killprocess was called. Kill pid = 1119  
-----

Killed

```

john@oho:~$ PID TTY      TIME CMD
      8 tty1      00:00:00 bash
     1127 tty1      00:00:00 a.out
     1132 tty1      00:00:00 sh
     1133 tty1      00:00:00 ps

```

-----  
Killprocess was called. Kill pid = 1127  
-----