**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 10

**(Spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 5/23/2024 |
| Course Code: | CSL - 320 | Max Marks: 20 |
| Faculty’s Name: | Abdullah |  |

Name:\_AFFAN AHMAD \_\_\_\_\_\_ Enroll No: \_03-134221-003\_\_\_

## Objective(s) :

To study about Signal Handling. Use top, ps and Kill commands.

## Lab Tasks :

**Task 1 :**Write the output for top and ps. Differentiate between the both terms.

**Task 2:** Write the output for the use of aux with ps for the firefox program.

**Task 3 :** Write the output of program for Kill Signal for firefox.

## Task 4.1 : Write a C program with an infinite loop and a custom signal handler to handle the interrupt signal (Ctrl+C).

**Task 4.2:** Write a program in C with an infinite loop and a custom signal handler to handle at least kill -15 (SIGTERM) and kill -9 (SIGKILL). Send both these signals (kill -9 and -15) using your running process’s PID.

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 5 |  |  |
| 2. | 5 |  |  |
| 3. | 5 |  |  |
| 4. | 5 |  |  |
| **Total** | **20** |  | **Signature** |

**Note : Attempt all tasks and get them checked by your Lab. Instructor**

# Lab 10: Signal Handling

**Objective(s):**

* To understand signal handling.

**Tool(s) used:**

Ubuntu, VIM Editor

**SIGNAL HANDLING**

**Top Command:** Top command displays processor activity of your Linux box and alsodisplays tasks managed by kernel in real-time. It'll show processor and memory are being used and other information like running processes.

**Output:**

## 



**Ps Command:** Report a snapshot of the current processes. Now to select a single requiredprocess we use aux along with ps.

**aux Command:**

a = show processes for all users

u = display the process's user/owner

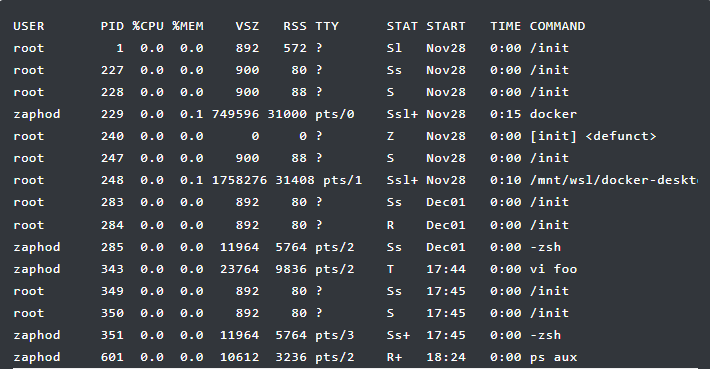
x = also show processes not attached to a terminal

**To Search a Particular Process:**

Open Firefox in background. To search its details, type:

**ps aux**

**Output:**



**Sending a Kill signal to a process:**

To kill a process many commands are used. Two of them are Kill -15 and Kill -9.

**Kill -15** sends a signal called SIGTERM. It kills the process gracefully. It means thatSIGTERM gives time to the process to save its state and also give it the option to ignore the signal. Kill and Kill -15 are same.

**Kill -9** sends a signal call SIGKILL. It kills the process ruthlessly giving it no time to save.Actually the process isn’t even made aware of the SIGKILL signal since the signal goes straight to the kernel init. At that point, init will stop the process. The process never gets the opportunity to catch the signal and act on it.

Now the send the signal using it’s PID like this and the process i.e firefox will be killed.

## Task 4.1 : Write a C program with an infinite loop and a custom signal handler to handle the interrupt signal (Ctrl+C).

#include<stdio.h>

#include<signal.h>

// Ctrl-C at keyboard

void handle\_sigint(int sig){

printf("Caught signal %d\n", sig);

}

int main( ){

signal(SIGINT, handle\_sigint);

while (1){

printf("hello world\n");

sleep(1);

}

return 0;

}

## In C++:

#include <iostream>

using namespace std;

#include <csignal>

#include <unistd.h>

// Ctrl-C at keyboard

void handle\_sigint(int sig){

cout << "Caught signal " << sig << std::endl;

}

int main() {

signal(SIGINT, handle\_sigint);

while (true) {

cout << "hello world" << std::endl;

sleep(1);

}

return 0;

}

**Output**

hello world

hello world

hello world

hello world

hello world

^CCaught signal 2

### Task 4.2 : Write a program in C with an infinite loop and a custom signal handler to handle at least kill -15 (SIGTERM) and kill -9 (SIGKILL). Send both these signals (kill -9 and -15) using your running process’s PID.

**Algorithm:**

* Write an infinite loop in C
* Include header file signal.h
* Call signal(signal\_name, your\_custom\_signal\_Handler) to catch the signal you will send.
* Write your custom signal Handler function below main and write its prototype before main.
* Compile by typing “gcc –o filename filename.c” in shell.
* Then execute it by “./filename”

**Solution:**

## #include <iostream>

## #include <csignal>

## #include <cstdlib>

## #include <unistd.h>

## // Function prototype for the custom signal handler

## void handle\_sigterm(int sig);

## int main() {

## // Register the signal handler for SIGTERM

## std::signal(SIGTERM, handle\_sigterm);

## // Infinite loop

## while (true) {

## std::cout << "Running... (PID: " << getpid() << ")" << std::endl;

## sleep(1);

## }

## return 0;

## }

## // Custom signal handler for SIGTERM

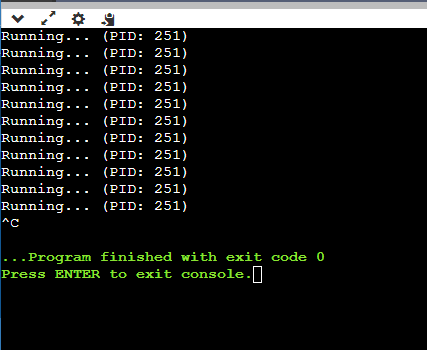
## void handle\_sigterm(int sig) {

## std::cout << "Caught signal " << sig << " (SIGTERM), exiting gracefully..." << std::endl;

## std::exit(0);

## }

## Output:



**Sending Signal:**

**SIGTERM:** Use process pid and send signal. Process will catch it. Since the infinite loop isrunning the process will ignore the signal as it is busy and it has the option to ignore.

Caught signal 15 (SIGTERM), exiting gracefully...

**SIGKILL:** Now send SIGKILL and kill it. You can see that the process has stoppedexecution and has been killed. Also it couldn’t catch the signal SIGKILL.

## [Process terminates]