

PRACTICAL 3

Aim- 1. Adam is working in an IT company. He has been given a task to reduce the load of a system by killing some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

1. Kill processes by name

```
MINGW64:/c/Users/91942/OneDrive/Desktop
91942@ANUSA MINGW64 ~/OneDrive/Desktop (main)
$ cmd.exe /c "tasklist | findstr notepad"
Microsoft Windows [Version 10.0.26100.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91942\OneDrive\Desktop>cmd.exe /c "taskkill /IM notepad.exe"
SUCCESS: Sent termination signal to the process "Notepad.exe" with PID 28600.
```

2. Kill a process based on the process name

```
C:\Users\91942\OneDrive\Desktop>taskkill /IM chrome.exe /F
SUCCESS: The process "chrome.exe" with PID 22804 has been terminated.
SUCCESS: The process "chrome.exe" with PID 25744 has been terminated.
SUCCESS: The process "chrome.exe" with PID 22084 has been terminated.
SUCCESS: The process "chrome.exe" with PID 38108 has been terminated.
SUCCESS: The process "chrome.exe" with PID 33352 has been terminated.
SUCCESS: The process "chrome.exe" with PID 32168 has been terminated.
SUCCESS: The process "chrome.exe" with PID 28812 has been terminated.
SUCCESS: The process "chrome.exe" with PID 3372 has been terminated.
SUCCESS: The process "chrome.exe" with PID 18760 has been terminated.

C:\Users\91942\OneDrive\Desktop>
```

3. Kill a single process at a time with the given process ID

```
C:\Users\91942\OneDrive\Desktop>tasklist | findstr chrome
chrome.exe          34548 Console          10    2,12,616 K
chrome.exe          34008 Console          10    11,308 K
chrome.exe          31444 Console          10    1,20,512 K
chrome.exe          33828 Console          10    46,036 K
chrome.exe          33260 Console          10    23,404 K
chrome.exe          1780  Console          10    73,560 K
chrome.exe          25620 Console          10    97,924 K
chrome.exe          22252 Console          10    1,10,184 K
chrome.exe          3296  Console          10    78,644 K
chrome.exe          22504 Console          10    23,732 K

C:\Users\91942\OneDrive\Desktop>taskkill /PID 34548
SUCCESS: Sent termination signal to the process with PID 34548.
```

2. Write a program for process creation using C

▪ Orphan Process

The screenshot shows a terminal window titled "orphan.c" containing the following C code:

```
GNU nano 8.7
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t pid = fork();

    if (pid > 0) {
        // Parent process
        printf("Parent process exiting...\n");
        printf("Parent PID: %d\n", getpid());
    } else if (pid == 0) {
        // Child process
        sleep(5);
        printf("Child process running...\n");
        printf("Child PID: %d\n", getpid());
        printf("New Parent PID (init/systemd): %d\n", getppid());
    } else {
        printf("Fork failed\n");
    }
    return 0;
}
```

The terminal window has a menu bar at the bottom with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, and Redo.

OUTPUT:

The screenshot shows a terminal window with the following session:

```
91942@ANUSHA MSYS ~
$ nano orphan.c

91942@ANUSHA MSYS ~
$ gcc orphan.c -o orphan

91942@ANUSHA MSYS ~
$ ./orphan
Parent process exiting...
Parent PID: 482

91942@ANUSHA MSYS ~
$ Child process running...
Child PID: 483
New Parent PID (init/systemd): 1
$ |
```

▪ Zombie Process

The screenshot shows a terminal window with the title "zombie.c". The file contains C code for a zombie process example. The code includes includes for stdio.h, unistd.h, and sys/types.h. It defines a main() function that forks. If the child process (pid == 0) exits immediately, printing "child exiting\n". Otherwise, the parent sleeps for 10 seconds, printing "Parent running\n". The terminal window also shows various menu options at the bottom.

```
GNU nano 8.7
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t pid = fork();
    if (pid == 0) {
        // Child process exits immediately
        printf("child exiting\n");
    } else {
        // Parent sleeps, does not call wait()
        sleep(10);
        printf("Parent running\n");
    }
    return 0;
}
```

File menu: Help, Write Out, Where Is, Cut, Paste, Execute, Location, Go To Line, Set Mark, To Bracket, Copy, where Was, Previous, Next, Back, Forward, Prev word, Next word.

OUTPUT:

The screenshot shows a terminal window with the title "MSYS ~". The user runs "nano zombie.c", compiles it with "gcc", and executes it with "./zombie". The output shows the child process exiting immediately and the parent process running for 10 seconds before exiting.

```
91942@ANUSHA MSYS ~
$ nano zombie.c

91942@ANUSHA MSYS ~
$ gcc zombie.c -o zombie

91942@ANUSHA MSYS ~
$ ./zombie
Child exiting
Parent running

91942@ANUSHA MSYS ~
$ |
```

3. Create the process using fork () system call.

- Child Process creation

- Parent process creation

- PPID and PID

The screenshot shows a terminal window with the title "GNU nano 8.7". The file name "Fork.c" is displayed at the top right. The code in the editor is:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
    } else if (pid == 0) {
        // Child process
        printf("Child Process\n");
        printf("PPID : %d\n", getppid());
    } else {
        // Parent process
        printf("Parent Process\n");
        printf("PID : %d\n", getpid());
        printf("Child PID: %d\n", pid);
        wait(NULL);
    }
    return 0;
}
```

The terminal window has a menu bar with "File", "Edit", "Search", "View", "Insert", "Format", "Tools", and "Help". Below the menu is a toolbar with icons for "Exit", "Write Out", "Where Is", "Replace", "Cut", "Paste", "Execute", "Justify", "Location", "Go To Line", "Undo", "Redo", "Set Mark", "To Bracket", "Copy", "Where Was", "Next", "Back Forward", and "Prev Word". A status bar at the bottom indicates "Read 30 lines".

OUTPUT:

The screenshot shows a terminal window with the title "M ~". The session starts with:

```
91942@ANUSHA MSYS ~
$ nano Fork.c
```

Then it compiles the code:

```
91942@ANUSHA MSYS ~
$ gcc Fork.c -o Fork
```

Finally, it runs the program:

```
91942@ANUSHA MSYS ~
$ ./Fork
Parent Process
Child Process
PID      : 511
Child PID: 512
PID      : 512
PPID     : 511
```