|  |  |  |  |
| --- | --- | --- | --- |
| **Fr. Conceicao Rodrigues College of Engineering Department of Computer Engineering** | | | |
| **Student’s Roll No** | **9603** | **Students Name** | **Zane Falcao** |
| **Date of Performance** |  | **SE Computer – Div** | **A** |

**Aim:** To study Process and File Management System Calls

**Lab Outcome:**

**CSL403.1: Demonstrate basic Operating system Commands, Shell scripts, System Calls and API wrt Linux.**

**Problem Statements:**

1. Process related System Calls

a)Create a child process in Linux using the fork system call. From the child process obtain the process ID of both child and parent by using getpid and getppid system call.

#include <stdio.h>  
#include <sys/types.h>  
#include <unistd.h>  
void forkexample()  
{  
if (fork() == 0)  
printf("Hello from Child!\n");  
else  
printf("Hello from Parent!\n");  
}  
int main()  
{int pid;  
pid = fork();  
if (pid == 0)  
printf("parent process id:%d\n",getpid());  
else  
printf("child process id:%d\n",getppid());  
forkexample();  
return 0;  
}

OUTPUT:

universe@dell9:~/Desktop/9603$ gcc fork.c  
universe@dell9:~/Desktop/9603$ ./a.out  
child process id:4354  
parent process id:4376  
Hello from Parent!  
Hello from Child!  
Hello from Child!  
Hello from Parent!

b) Explore wait and waitpid before termination of process.

//Demonstrate the working of wait

#include<stdio.h>  
#include<stdlib.h>  
#include<sys/wait.h>  
#include<unistd.h>  
int main()  
{  
pid\_t cpid;  
if (fork()== 0)  
exit(0);  
else  
cpid = wait(NULL); /\* reaping parent \*/  
printf("Parent pid = %d\n", getpid());  
printf("Child pid = %d\n", cpid);  
return 0;  
}

OUTPUT:

universe@dell9:~/Desktop/9603$ gcc wait.c  
universe@dell9:~/Desktop/9603$ ./a.out  
Parent pid = 4402  
Child pid = 4403

c) Explain ps command and output in detail. What is Zombie and Orphan Process? Show the output.

1] ps: The ps command is used to view currently running processes on the system. The ps command may display different results for different systems because it displays information about the currently running process.

zane@DESKTOP-LJHRUE9:~$ ps

PID TTY TIME CMD

10 tty1 00:00:00 bash

43 tty1 00:00:00 ps

PID is the process ID of running command

TTY is the type of terminal where the current command is running

TIME tells how much time is used by CPU to the run the process

CMD is the current command

2] ps -aux: This command filters the result by CPU or memory usage. It helps you in determining how much memory is used or how much CPU space is used by a process.

zane@DESKTOP-LJHRUE9:~$ ps -aux

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

zane 10 0.5 0.0 14248 3800 tty1 S 21:15 0:00 -bash

zane 45 0.0 0.0 15828 2104 tty1 R 21:16 0:00 ps -ux

3] ps -ef: To display all currently running processes in full format on a system.

zane@DESKTOP-LJHRUE9:~$ ps -ef

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 21:15 ? 00:00:00 /init

root 9 1 0 21:15 tty1 00:00:00 /init

zane 10 9 0 21:15 tty1 00:00:00 -bash

zane 46 10 0 22:01 tty1 00:00:00 ps -ef

UID: the user ID of the user who owns the process

PID: the process ID of the process

PPID: the process ID of the parent process

C: the CPU utilization of the process, as a percentage

STIME: the time at which the process was started

TTY: the terminal associated with the process

TIME: the CPU time used by the process

CMD: the command that was used to start the process, including any command-line arguments passed to the process

4] ps-L: List all threads for a particular process

zane@DESKTOP-LJHRUE9:~$ ps -L

PID LWP TTY TIME CMD

10 10 tty1 00:00:00 bash

48 48 tty1 00:00:00 ps

5] ps -f -u username: It displays all the process of the username

zane@DESKTOP-LJHRUE9:~$ ps -f -u zane

UID PID PPID C STIME TTY TIME CMD

zane 10 9 0 21:15 tty1 00:00:00 -bash

zane 57 10 0 22:03 tty1 00:00:00 ps -f -u zane

6] ps-a: Views processes not associated with a terminal

zane@DESKTOP-LJHRUE9:~$ ps -a

PID TTY TIME CMD

10 tty1 00:00:00 bash

59 tty1 00:00:00 ps

7] ps-x: This command will show all the processes that are owned by the user that is running the ps command.

zane@DESKTOP-LJHRUE9:~$ ps -x

PID TTY STAT TIME COMMAND

10 tty1 S 0:00 -bash

60 tty1 R 0:00 ps -x

8] ps-eLf: ps -eLf outputs info for every running process (-e), in long format (-l - shows extra columns of useful information), and full format (-f), which shows more info as well, including command line arguments.

zane@DESKTOP-LJHRUE9:~$ ps -eLf

UID PID PPID LWP C NLWP STIME TTY TIME CMD

root 1 0 1 0 2 21:15 ? 00:00:00 /init

root 1 0 7 0 2 21:15 ? 00:00:00 /init

root 9 1 9 0 1 21:15 tty1 00:00:00 /init

zane 10 9 10 0 1 21:15 tty1 00:00:00 -bash

zane 61 10 61 0 1 22:09 tty1 00:00:00 ps -eLf

9] ps-U root -u root u: This command is used to reveal all processes run by the root user

zane@DESKTOP-LJHRUE9:~$ ps -U root -u root u

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.0 8960 412 ? Ssl 21:15 0:00 /init

root 9 0.0 0.0 9308 236 tty1 Ss 21:15 0:00 /init

10] ps -T: Views all the processes associated with the terminal

zane@DESKTOP-LJHRUE9:~$ ps -T

PID SPID TTY TIME CMD

10 10 tty1 00:00:00 bash

64 64 tty1 00:00:00 ps

11] ps -C process\_name: This command is used to search the PID of a process

zane@DESKTOP-LJHRUE9:~$ ps -C bash

PID TTY TIME CMD

10 tty1 00:00:00 bash

12] ps -fp PID: This command is used to display the process by their given PID

zane@DESKTOP-LJHRUE9:~$ ps -fp 10

UID PID PPID C STIME TTY TIME CMD

zane 10 9 0 21:15 tty1 00:00:00 -bash

ZOMBIE PROCESS:

A process that has finished the execution but still has an entry in the process table to report to its parent process is known as a zombie process.

CODE:

#include <stdlib.h>  
#include <sys/types.h>  
#include <unistd.h>

int main()

{  
pid\_t child\_pid = fork();  
if (child\_pid > 0)  
sleep(50);  
else  
exit(0);  
return 0;  
}

OUTPUT:

universe@dell9:~/Desktop/9603$ gcc zombie.c  
universe@dell9:~/Desktop/9603$ ./a.out

universe@dell9:~/Desktop/9603$ ps -aux | egrep “Z|defunct”

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

universe    4410   0.0   0.0     0     0 pts/0    Z+   10:53   0:00 [a.out] <defunct>

ORPHAN PROCESS:

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.

CODE:

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{

pid\_t p;

p=fork();

if(p==0)

{ sleep(3); // child process goes to sleep and the parent process terminates

printf(“I am child having PID %d\n”,getpid());

printf(“My parent PID is %d\n”,getppid());

}

else

{

printf(“I am parent having PID %d\n”,getpid());

printf(“My child PID is %d\n”,p)

}

}

OUTPUT:

universe@dell9:~/Desktop/9603$ gcc orphan.c  
universe@dell9:~/Desktop/9603$ ./a.out

I am parent having PID 138

My child PID is 90

universe@dell9:~/Desktop/9603$ I am child having PID 90

My parent PID is 120

d) Explain fork(), getpid(), getppid(),wait() and waitpid() with syntax.

(2) File related system calls

a) Program to copy contents of one file (source) to another file (destination). Finally displaying contents of destination file.

b) 2. Explain creat(), open(), close(), read() and write() with syntax.

|  |  |  |  |
| --- | --- | --- | --- |
| **On time Submission(2)** | **Knowledge of Topic(4)** | **Implementation and Demonstraion(4)** | **Total (10)** |
|  |  |  |  |
| **Signature of Faculty** |  | **Date of Submission** |  |