Instituto Politécnico Nacional

Escuela Superior de Cómputo

Sistemas Operativos

Laboratorio de Sistemas Operativos

**Práctica No.4**

**“Ejecución de procesos con exec()”**

Grupo: 2CM8

Ayona Lopez Eugenio Milton

*Profesor: Juárez Méndez Ana Belem*

2

OBJETIVO

PRACTICA NO.4

*“EJECUCIÓN DE PROCESOS CON EXEC()”*

Aplicar los conocimientos sobre la ejecución de procesos desde otros procesos mediante el uso de la herramienta exec que nos proporciona el Sistema Operativo UNIX- LINUX.

DESARROLLO

*I. EJERCICIO 1*

Parte A) Realiza un programa en c que ejecute la orden de cualquier comando del intérprete de comandos de la terminal. El comando para ejecutar debe pasarse como argumentos en el main.

CODIGO LENGUAJE C

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

int main(int argc,char \*argv[])

{

//Arreglo que guarda los argumentos char \*argumentos[argc];

if(argc<2){

printf("Por favor,ingrese un argumento\n");

}else{

//guardar los argumentos ingresados

for (int i=1;i<=argc;i++)

{

argumentos[i-1]=argv[i];

}

argumentos[argc]=0;

execvp(argumentos[0],argumentos);

}

exit(0);

}

De acuerdo con el código anterior, se tomo como base para incluir la siguiente parte del ejercicio, de esta forma, en el siguiente programa se crea un proceso que ejecuta las instrucciones de otro archivo que contienen a las líneas anteriores con el uso de la función execvp().

3

Parte B) Realiza un programa en c que pida al usuario que introduzca cualquier comando del interprete de comandos de la terminal. El programa debe leer el comando y crear un proceso que mande a ejecutar el programa que realizaste en la Parte a con el comando que introdujo el usuario.

El programa principal( proceso padre) debe esperar a que termine la ejecución su hijo (programa que realizaste en la Parte a) y cuando termine, volver a pedir al usuario que introduzca otro comando. El proceso padre (programa principal) deberá dejar de pedir comandos al usuario, cuando el usuario introduzca exit.

CODIGO LENGUAJE C

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

int pid=1;

int main(int argc,char \*argv[])

{

int status;

int i=0;

char cadena[50];

char \*final[50];

char delimitador[] = " ";

char \*varprog="./a";

system("clear");

do

{

NULL

i=0;

final[i]=(char \*)varprog;

i++;

//Inicio del programa a ejecutar printf("Ingrese un comando> "); fflush(stdin);

gets(cadena);

char \*token = strtok(cadena, delimitador);

if(token != NULL){

while(token != NULL){

// Sólo en la primera pasamos la cadena; en las siguientes pasamos

final[i]=(char \*)token;

token = strtok(NULL, delimitador);

i++;

}

}

//Delimitacion de los argumentos final[i]=0;

if(pid!=0)

{

if(pid==-1)

4

{

perror("\nError en el uso de fork()");

exit(pid);

}else{

pid=fork();

wait(&status);

}

}else{

break;

}

if(pid==0){

printf("%d\n",execvp("./a",final));

exit(0);

}

}while(strcmp(cadena,"exit")!=0);

printf("Ejecucion de programa principal ha terminado.\n");

return 0;

}

RESULTADOS

Resultado probando con la ejecución del comando ls, ls con parámetros -a -l y el comando cal con parámetros -m4.



5

Resultado probando con el comando cal sin parámetros, de la misma forma que el comando uname sin parámetros y finalmente llevando a cabo la ejecución de la compilación de este mismo archivo en c a través del uso del comando gcc con todos sus parámetros.



Como se logra observar en la última línea de instrucciones, al colocar el comando exit, este termina con la ejecución del programa principal.