UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5314	Sistemas Operativos

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
3	NOMBRE DE LA PRÁCTICA	Llamas al sistema exec() y system()	2

1. INTRODUCCIÓN

Las llamadas al sistema proporcionan la interfaz entre el sistema operativo y un programa en ejecución. Linux permite efectuar llamadas al sistema directamente desde un programa realizado en un lenguaje de alto nivel, en particular en lenguaje C, en cuyo caso las llamadas se asemejan a llamadas a funciones, tal y como si estuvieran definidas en una biblioteca estándar. En esta práctica trabajaremos con las funciones exec() y system() las cuales permiten realizar llamadas al sistema.

2. OBJETIVO (COMPETENCIA)

En esta práctica el alumno se familiarizará con las llamadas al sistema exec( ) y system( ) pudiendo deducir de los ejemplos la diferencia en el funcionamiento de cada una de ellas.

3. FUNDAMENTO

Cuando se ejecuta un programa se copia el ejecutable en una imagen de programa (imagen de carga) en la memoria principal y se le asigna un espacio de direcciones. Cada programa en ejecución es un proceso que es manejado por el sistema mediante una estructura de datos a la que se accede a través de un entero denominado identificador de proceso (*pid*). La familia de llamadas **exec ( )**, reemplaza la imagen del proceso actual con una nueva imagen de carga referente al archivo nombrado en el primer parámetro. En el caso más general, **exec** tiene tres parámetros: el nombre del archivo por ejecutar, un apuntador al arreglo de argumentos y un apuntador al arreglo del entorno. Típicamente, después de una bifurcación (*fork*), uno de los dos procesos emplea la llamada **exec**, para reemplazar su espacio de memoria virtual con un nuevo programa. Esta llamada carga en memoria un archivo binario (destruyendo el contenido en memoria del programa que contiene la llamada, y comienza su ejecución. Se dispone de varios procedimientos de biblioteca, *execl, execlp, execle, exect, execv, execve,* y *execp,* para que los parámetros sean omitidos o especificados de varias formas.

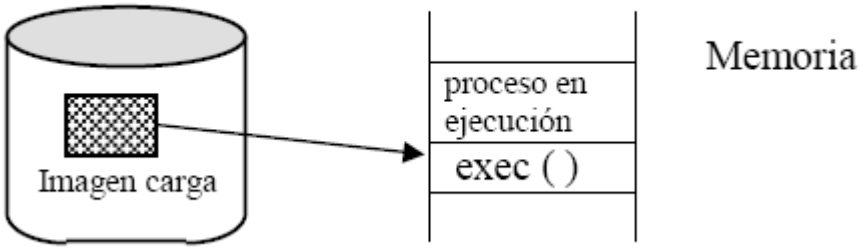

Figura 1

En la figura 1 se observa que el proceso que hace la llamada **exec**, pierde el control del procesador para ejecutar el código correspondiente al archivo cargado. Las líneas de código escritas en un programa que vienen justamente detrás de **exec** no se ejecutan, salvo si hay algún error en la carga.

Sintaxis de la función excel()

int = execl(nombre_absoluto, arg0,arg1,...,0);

Donde:
nombre_absoluto: es el *pathname* absoluto donde reside la imagen de carga del fichero a ejecutar.
arg0: es el nombre del fichero que se quiere ejecutar.

Los parámetros para el archivo que se quiere ejecutar vienen especificados desde *arg1* a *argN*, siendo *argN* el último argumento que se pasa, terminando la llamada siempre con un "0, para indicar que el argumento anterior es el último.

Programa ejemplo que muestra el uso de execl

```
/* exec.c - Listar los procesos del usuario usando exec. */
#include <stdio.h>
#include <unistd.h>

int main ()
   {
   int  salida;/* Salida del comando */

   printf ("Ejemplo de exec \n");
   execl ("/bin/ps", "ps", "-fu", getenv ("USER"), 0);
   printf ("Salida del comando: %d\n", salida);
exit (0); }
```

La salida del programa será la siguiente:

Ejemplo de exec
UID PID PPID C STIME TTY TIME CMD
lopez 29395 29393 0 Apr29 ? 00:00:00 [sshd]
lopez 29396 29395 0 Apr29 pts/1 00:00:00 -bash
lopez 5266 29396 0 12:39 pts/1 00:00:00 ps -fu lopez

system( )

La subrutina system( ) llamada a un intérprete para ejecutar una orden El proceso espera a que finalice la ejecución de la subrutina y devuelve la salida del programa ejecutado.

Sintaxis de la subrutina system

```
int = system(cadena);
```

Programa ejemplo que muestra el uso de system( )

```
/* system.c - Listar los procesos del usuario usando system. */
#include <stdio.h>
#include <stdlib.h>
int main ()
{
   int  salida;     /* Salida del comando */
   char comando[100];       /* Comando a ejecutar */
   printf ("Ejemplo de system.\n");
   sprintf (comando, "/bin/ps -fu %s", getenv ("USER"));
   salida = system (comando);
   printf ("Salida del comando: %d\n", salida);
   exit (0);
}
```

La salida del programa será la siguiente:

Ejemplo de system.
UID PID PPID C STIME TTY TIME CMD
lopez 29395 29393 0 Apr29 ? 00:00:00 [sshd]
lopez 29396 29395 0 Apr29 pts/1 00:00:00 -bash
lopez 5285 29396 0 12:42 pts/1 00:00:00 ./procesos1
lopez 5286 5285 2 12:42 pts/1 00:00:00 /bin/ps -fu lopez
Salida del comando: 0

4. PROCEDIMIENTO (DESCRIPCIÓN)

Formatos para prácticas de laboratorio

A) EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con sistema operativo Linux.	Práctica 3 del Manuel manual de Sistemas Operativos1.- Elabora un programa en C bajo Linux

utilizando la función exec() que funciones como un MINISO. Tu MINISO deberá admitir las siguientes ordenes:

Listar (ls)
Fecha (date)
Renombrar (mv)
Borrar (rm)
Limpiar(ls)

Cuando el programa inicie deberá mostrar al usuario el siguiente prompt:

MINISO]$

Cuando el usuario escriba alguna de las ordenes permitidas deberá mostrar el resultado de la ejecución de la orden y volver a mostrar el prompt hasta que el usuario escriba la palabra Salida.

Ejemplo 1:

MINISO]$ Borrar MiArchivo
MINISO]$ Salida

Ejemplo 2:

MINISO]$ Renombrar MiArchivo TuArchivo
MINISO]$ Salida

2.- Elabore el mismo MINISO planteado en el paso 1, pero ahora utilizando la función system().
3.- ¿Cuál es la diferencia entre las funciones exec() y system()?.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

Al finalizar la práctica el alumno conocerá la diferencia entre las llamadas a las funciones system() y exec().

6. ANEXOS

7. REFERENCIAS

http://yaqui.mxl.uabc.mx/~so