Reporte de la práctica 3

Hernández Tapia Luis Enrique

Sistemas Operativos
Grupo 2CM7
Profesor: Montes Casiano Hermes Francisco
27 de febrero de 2018

1. Introducción

Todos los programas ejecutables están en disco, deben ser cargados en memoria para ejecutarse y, por consiguiente, convertirse en procesos. Para administrar los procesos, el sistema operativo LINUX identifica cada uno mediante su PID.

2. Desarrollo

Se nos pide escribir el siguiente programa 3-1 y guardarlo como programa3-1.c

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

int main(int argc, char **argv){
        printf("Identificador de usuario: %d\n", getuid());
        return 0;
}
```

Posteriormente construir y ejecutar

A continuación se nos pide realizar el programa 3-2

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

int main(int argc, char **argv){
        pid_t id_proceso;
        pid_t id_padre;

        id_proceso = getpid();
        id_padre = getppid();

        printf("Identificador de proceso: %d\n", id_proceso);
        printf("Identificador del proceso padre: %d\n", id_padre
           );

        sleep(20);
        return 0;
}
```

Para finalizar el programa 3-3

```c
#include <stdlib.h>
#include <stdio.h>

extern char **environ;

int main(int argc, char *argv[]){
        int j;

        printf("Las variables de entorno para %s son\n", argv
           [0]);

        for(j=0; environ[j] != NULL; j++){
                printf("environ[%d] = %s\n", j , environ[j]);
        }
        return 0;
}
```

3. Pruebas

Pregunta 3.1 - Investigue con ayuda de man cuál es la utilidad de la función getuid
R: Devuelve la identificación de usuario real del proceso de llamada.

Pregunta 3.2 - Investigue con ayuda de man cuál es la utilidad de la función getpid y getppid.
R: getpid(): Devuelve el ID de proceso (PID) del proceso de llamada.
getppid(): Devuelve el ID de proceso del padre de la llamada proceso.

Pregunta 3.3 - Si se ejecuta varias veces el programa, ¿por qué el identificador del padre es siempre el mismo (recomendación: utilice el comando postree de LINUX)?
Analizando con

```
$ pstree -p
```

observamos que su padre siempre será el bash.

Pregunta 3.4 - ¿En qué archivo se encuentra definida la variable environ?
en etc/environment y contiene el PATH

Pregunta 3.5 - ¿Qué significa las siguientes variables de entorno: HOME, LOGNAME, PATH, TERM, PWD (véase con man)?
HOME: muestra la ubicación del usuario en el disco
LOGNAME: imprime el nombre del usuario que ha iniciado sesión.
PATH: /usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin. La ruta del PATH
TERM: terminal por defecto
PWD: imprime el nombre del directorio actual

Ejercicio 3.3 - Usando la función getenv() (busque su uso con man), modifique este programa para que imprima únicamente el valor de la variable HOME.
Resultado:

```
#include <stdlib.h>
#include <stdio.h>

extern char **environ;

int main(int argc, char *argv[]){
        int j;

        printf("Las variables de entorno para %s son\n", argv
           [0]);
        /*
        for(j=0; environ[j] != NULL; j++){
                printf("environ[%d] = %s\n", j , environ[j]);
        }*/

        printf("HOME: %s/\n",getenv("HOME"));
        return 0;
}
}
```

4. Conclusiones

La importancia de saber cómo manejar los procesos, los estados que pueden tener, cómo se organizan y las prioridades que pueden tener es crucial para un administrador de sistemas, y cualquiera que desee saber, en un momento dado, si la CPU de su equipo comienza a sobre-utilizarse, o incluso de que haya muchos procesos que ocupen la memoria y permanezcan de forma fija en modo estático.