Procesos con Linux

Objetivos

- Introducir los conceptos de proceso de manera práctica.
- Incorporar la participación de los alumnos en durante el avance de la clase.

Introducción

1. Abrir una terminal en Linux
2. Mostrar los procesos que están corriendo en esa terminal
3. Indicar el nombre del proceso y el id
4. Abrir otra terminal y visualizar todos los procesos que se están ejecutando e identificar el estado de los procesos en ese momento.
5. Elegir un procesos, identificar el PID y acceder al PCB
6. Mostrar el árbol de procesos

Ejercicio 1 – fork

1. Crear un proceso con la llamada fork() como réplica del proceso padre. Usar el código propuesto.
2. Compilar y ejecutar en línea de comando
3. Discutir los resultados obtenidos.
   Recordar que Fork() devuelve:
   - 0 al proceso creado(al hijo).
   - El PID del proceso creado (hijo) al padre.
   - -1 en caso de error.

Ejercicio 2 – exec

1. Crear un proceso cuya ejecución sea independiente de la del padre y concurrente con ella. El padre con la llamada wait() espera la finalización del hijo. Usar el código propuesto.
2. Compilar y ejecutar.
3. Analizar y discutir los resultados

Ejercicio 3 – getpid y getppid

1. Crear varios procesos como réplica del proceso padre. Usar los comandos getpid() y getppid() para identificar cuando se ejecuta el padre y cuando se ejecuta el hijo. Usar el código propuesto.
2. Compilar y ejecutar en línea de comando
3. Analizar el resultado de la ejecución. ¿Que ocurre con el orden de ejecución?

Creación de un proceso

Ejercicio 1 usando fork()

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

int main() {

int pid;
pid = fork(); //aqui recibimos lo que devuelve la funcion fork()
// ahora queremos saber si estamos en el proceso padre o el hijo
if ( pid > 0) {printf("Soy el proceso padre \n"); while(1);
             }
else {
      if (pid == 0) {
            printf("Soy el proceso hijo\n"); while(1);
      }else {
            printf("Error en la funcion fork()\n");
            }
    }
}
```

Ejercicio 2

crea un proceso con **fork()** y carga en memoria un **nuevo programa** usando **exec()**

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

int main()
{
    int  pid;
    pid = fork();       /* bifurca un proceso hijo*/
    if (pid < 0) { /* ocurre un error*/
        fprintf(stderr, "Fork Failed");
        exit(-1);
    } else if (pid == 0) { /* proceso hijo */
        execlp("/bin/ls", "ls", NULL);
    } else { /* proceso padre*/
        wait (NULL);   /* padre espera a que se complete el
hijo*/
        printf("Proceso padre   Child Complete. \n");
                exit(0);
            }
}
```

Ejercicio 3 usando Getpid() y Getppid()

/*crea tres procesos, espera con sleep 5 seg y finaliza los hijos y por último el padre*/

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

int main () {
int num;
pid_t pid;  // define tipo pid


for (num= 0; num< 3; num++) {
pid= fork();       // crea un proceso

printf ("Soy el proceso de PID %d, mi padre tiene %d de PID.\n",
getpid(), getppid());
if (pid== 0)
break;
}
if (pid== 0)
sleep(5);
else
for (num= 0; num< 3; num++)
printf ("Fin del proceso de PID %d.\n", wait (NULL));
return 0;
}
}
```