Sistemas Operativos

Formulario de auto-evaluación

Modulo 2. Sesión 5. Llamadas al sistema para gestión y control de señales.

Nombre y apellidos:

Antonio Miguel Morillo Chica

a) Cuestionario de actitud frente al trabajo.

El tiempo que he dedicado a la preparación de la sesión antes de asistir al laboratorio ha sido de 0 minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: ..... (si/no). En caso de haber contestado “no”, indica los motivos por los que no las has resuelto:

Sí, pero retomé la practicas días despues y tuve que volver a revisar las cosas.

2. Tengo que trabajar algo más los conceptos sobre:

Las señales y las máscaras.

3. Comentarios y sugerencias:

Puede que esta practica esté en un nivel más bajo pero debido a que esta semana desde el viernes me exigieron muchas practicas y entregas y no pude ponerme en ello hasta el último día, antes de la entrega.

b) Cuestionario de conocimientos adquiridos.

Mi solución al **ejercicio 2** ha sido:

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

int sighup = 0, sigint = 0,
   sigfpe = 0, sigpipe = 0,
   sigalrm = 0, sigusr1 = 0;

// Funcion para saber que señal se ha captado.
void handler (int signum){

  if (signum == SIGINT) {
    sigint++;
    printf("Señal SIGINT recibida\nNumero de veces que se ha recibido: %d\n", sigint);
  }

  if (signum == SIGHUP) {
    sighup++;
    printf("Señal SIGHUP recibida\nNumero de veces que se ha recibido: %d\n", sighup);
  }

  if (signum == SIGFPE) {
    sigfpe++;
    printf("Señal SIGFPE recibida\nNumero de veces que se ha recibido: %d\n", sigfpe);
  }

  if (signum == SIGPIPE) {
    sigpipe++;
    printf("Señal SIGPIPE recibida\nNumero de veces que se ha recibido: %d\n", sigpipe);
  }

  if (signum == SIGALRM) {
    sigalrm++;
    printf("Señal SIGALRM recibida\nNumero de veces que se ha recibido: %d\n", sigalrm);
  }

  if (signum == SIGUSR1) {
    sigusr1++;
    printf("Señal SIGUSR1 recibida\nNumero de veces que se ha recibido: %d\n", sigusr1);
  }
}

int main()
{
  struct sigaction sa;
```

```
  if(setvbuf(stdout,NULL,_IONBF,0))
    perror("\nError en setvbuf");

  // Establezco mi propio manejador. sa_handler capta las señales
  sa.sa_handler = handler;

  // Recojo los posbles errores.
  if (sigaction(SIGINT, &sa, NULL) == -1) {
    printf("Error en el manejador SIGINT");
    exit -1;
  }

  if (sigaction(SIGHUP, &sa, NULL) == -1) {
    printf("Error en el manejador SIGHUP");
    exit -1;
  }

  if (sigaction(SIGFPE, &sa, NULL) == -1) {
    printf("Error en el manejador SIGFPE");
    exit -1;
  }

  if (sigaction(SIGPIPE, &sa, NULL) == -1) {
    printf("error en el manejador");
    exit -1;
  }

  if (sigaction(SIGALRM, &sa, NULL) == -1) {
    printf("Error en el manejador SIGALRM");
    exit -1;
  }

  // Dejo SIGUSR1 para que de error y se finalice el proceso.

  while(1);
}
```

Mi solución a la **ejercicio 3** ha sido:

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

int main(){
  sigset_t new_mask;
  // inicializar la nueva mascara de señales (todas)
  sigfillset(&new_mask);
  // señal que se borra de la lista y que finaliza la suspension
  sigdelset(&new_mask, SIGUSR1);
  // esperar a cualquier señal excepto SIGUSR1
  sigsuspend(&new_mask);
}
```

Mi solución a la **ejercicio 4** ha sido:

Lo que hace el programa es bloquear la señal SIGTERM durante 10 segundos. L oque hace es crearse un conjunto con todas las señales y otra con solo SIGTERM. Posteriormente bloquea de todas las señales aquellas del conjunto que solo contine a SIGTERM de manera que durante un instante la señal SIGTERM no tendrá ningún efecto, a excepción de pasados 10 sec que se desbloquea.