PRACTICA 4

Created	@January 15, 2021 10:31 AM
Date	@October 26, 2020 → November 8, 2020
Tags	
Tema	

Tema i què es fa

Comunicación de procesos, signals entre un proceso y otro como método de comunicación.

Sessió i codis

▼ bucleInfinito.c

```
//Extremely hard code
int main(int argc, char **argv) {
  while(1);
}
```

▼ ejemplo_alarm2.c

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

void error_y_exit(char *msg,int exit_status)
{
    perror(msg);
    exit(exit_status);
}

/* ESTA VARIABLE SE ACCEDE DESDE LA FUNCION DE ATENCION AL SIGNAL Y DESDE EL MAIN */
int segundos=0;
/* FUNCION DE ATENCION AL SIGNAL SIGALRM */
void funcion_alarma(int s)
```

```
{
    if (s == SIGALRM) segundos=segundos+10;
    else {
        char buff[256];
        sprintf(buff, "ALARMA pid=%d: %d segundos\n",getpid(),segundos);
        write(1, buff, strlen(buff));
    }
}
int main (int argc,char * argv[])
{
    struct sigaction sa;
    sigset_t mask;

    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigaddset(&mask, SIGUSR1);
    sigprocmask(SIG_BLOCK,&mask, NULL);

    /* REPROGRAMAMOS EL SIGNAL SIGALRM */
    sa.sa_handler = &funcion_alarma;
    sa.sa_flags = SA_RESTART;
    sigfillset(&sa.sa_mask);

    if (sigaction(SIGALRM, &sa, NULL) < 0 || sigaction(SIGUSR1, &sa, NULL) < 0) error_y_exit("sigaction", 1);

    while (segundos<100)
    {
  alarm(10);  /* Programamos la alarma para dentro de 10 segundos */
        /* Nos bloqueamos a esperar que nos llegue un evento */
      sigfillset(&mask);
        sigdelset(&mask, SIGALRM);
      sigdelset(&mask, SIGINT);
        sigdelset(&mask, SIGUSR1);
        sigsuspend(&mask);
    }
    exit(1);
}
```

▼ ejemplo_alarm3.c

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

void error_y_exit(char *msg,int exit_status)
{
    perror(msg);
    exit(exit_status);
}

int segundos=0;
void funcion_alarma(int signal)
{
  char buff[256];
  segundos=segundos+10;
  sprintf(buff,"ALARMA pid=%d: %d segundos\n",getpid(),segundos);
  write(1, buff, strlen(buff) );
}

int main (int argc,char * argv[])
{
    struct sigaction sa;
    sigset_t mask;

    /* EVITAMOS QUE NOS LLEGUE EL SIGALRM FUERA DEL SIGSUSPEND */
    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigprocmask(SIG_BLOCK,&mask, NULL);
```

```
    int pid;
    while (segundos<100) {
      alarm(10);
        pid = fork();
        if (pid < 0) error_y_exit("fork", 1);
        else if (pid == 0) {
            /*
      //REPROGRAMAMOS EL SIGNAL SIGALRM
            sa.sa_handler = &funcion_alarma;
            sa.sa_flags = SA_RESTART;
            sigfillset(&sa.sa_mask);
            if (sigaction(SIGALRM, &sa, NULL) < 0) error_y_exit("sigaction", 1);
      */
      execlp("./bucleInfinito", "./bucleInfinito", NULL);
        }

        sigfillset(&mask);
        sigdelset(&mask, SIGALRM);
        sigdelset(&mask, SIGINT);
        sigsuspend(&mask);
    }
    exit(1);
}
```

▼ ejemplo_signal.c

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

int contador = 0;
int hijos = 0;

void error_y_exit(char* msg, int exit_status)
{
    perror(msg);
    exit(exit_status);
}

void trata_hijo(int s) {
  int pid, exit_code;
  char buff[256];
  while ((pid = waitpid(-1, &exit_code, WNOHANG)) > 0) {
    if (WIFEXITED(exit_code)) {
      int statcode = WEXITSTATUS(exit_code);
      sprintf(buff,"Termina el proceso %d com exit code %d\n",pid, statcode);
    }
    else {
      int signcode = WTERMSIG(exit_code);
      sprintf(buff,"Han matado al proceso %d antes de acabar alarm por el signal %d\n",pid, signcode);
    }
    write(1, buff, strlen(buff));
    hijos--;
    ++contador;
  }
}


void trata_alarma(int s)
{
}

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

```
{
    int pid, res;
    char buff[256];
    struct sigaction sa;
    sigset_t mask;

    /* Evitamos recibir el SIGALRM fuera del sigsuspend */

    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigprocmask(SIG_BLOCK, &mask, NULL);

    for (hijos = 0; hijos < 10; hijos++) {
        sprintf(buff, "Creando el hijo numero %d\n", hijos);
        write(1, buff, strlen(buff));

        pid = fork();
        if (pid == 0) /* Esta linea la ejecutan tanto el padre como el hijo */
        {

            sa.sa_handler = &trata_alarma;
            sa.sa_flags = SA_RESTART;
            sigfillset(&sa.sa_mask);
            if (sigaction(SIGALRM, &sa, NULL) < 0)
                error_y_exit("sigaction", 1);

            /* Escribe aqui el codigo del proceso hijo */
            sprintf(buff, "Hola, soy %d\n", getpid());
            write(1, buff, strlen(buff));

            alarm(2);
            sigfillset(&mask);
            sigdelset(&mask, SIGALRM);
            sigdelset(&mask, SIGINT);
            sigsuspend(&mask);

            /* Termina su ejecución */
            exit(0);
        } else if (pid < 0) {
            /* Se ha producido un error */
            error_y_exit("Error en el fork", 1);
        }
    }
    /* Esperamos que acaben los hijos */
  sa.sa_handler = &trata_hijo;
    sa.sa_flags = SA_RESTART;
    sigaction(SIGCHLD, &sa, NULL);
    while (hijos > 0);
    sprintf(buff, "Valor del contador %d\n", contador);
    write(1, buff, strlen(buff));
    return 0;
}
```

▼ ejemplo_signal2.c

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

int contador = 0;
int hijos = 0;

void error_y_exit(char* msg, int exit_status)
{
    perror(msg);
    exit(exit_status);
}
```

```
void trata_hijo(int s) {
  int pid, exit_code;
  char buff[256];
  while ((pid = waitpid(-1, &exit_code, WNOHANG)) > 0) {
    if (WIFEXITED(exit_code)) {
      int statcode = WEXITSTATUS(exit_code);
      sprintf(buff,"Termina el proceso %d com exit code %d\n",pid, statcode);
    }
    else {
      int signcode = WTERMSIG(exit_code);
      sprintf(buff,"Han matado al proceso %d antes de acabar alarm por el signal %d\n",pid, signcode);
    }
    write(1, buff, strlen(buff));
    hijos--;
    ++contador;
  }
}


void trata_alarma(int s)
{
}

int main(int argc, char* argv[])
{
    int pid, res;
    char buff[256];
    struct sigaction sa;
    sigset_t mask;
  int pid_vec[10];

    /* Evitamos recibir el SIGALRM fuera del sigsuspend */

    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigprocmask(SIG_BLOCK, &mask, NULL);

    for (hijos = 0; hijos < 10; hijos++) {
        sprintf(buff, "Creando el hijo numero %d\n", hijos);
        write(1, buff, strlen(buff));

        pid = fork();
        if (pid == 0) /* Esta linea la ejecutan tanto el padre como el hijo */
        {
            sa.sa_handler = &trata_alarma;
            sa.sa_flags = SA_RESTART;
            sigfillset(&sa.sa_mask);
            if (sigaction(SIGALRM, &sa, NULL) < 0)
                error_y_exit("sigaction", 1);

            /* Escribe aqui el codigo del proceso hijo */
            sprintf(buff, "Hola, soy %d\n", getpid());
            write(1, buff, strlen(buff));

            alarm(2);
            sigfillset(&mask);
            sigdelset(&mask, SIGALRM);
            sigdelset(&mask, SIGINT);
            sigsuspend(&mask);

            /* Termina su ejecución */
            exit(0);
        } else if (pid < 0) {
            /* Se ha producido un error */
            error_y_exit("Error en el fork", 1);
        } else pid_vec[hijos] = pid;
    }
    /* Esperamos que acaben los hijos */
  sa.sa_handler = &trata_hijo;
    sa.sa_flags = SA_RESTART;
    sigaction(SIGCHLD, &sa, NULL);
  for (int i = 0; i < 10; ++i) kill(pid_vec[i], SIGUSR1);
```

```
    while (hijos > 0);
    sprintf(buff, "Valor del contador %d\n", contador);
    write(1, buff, strlen(buff));
    return 0;
}
```

▼ eventos.c

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

int contador = 0;

void trata(int s)
{
    if (s == SIGALRM) contador += 1;
    if (s == SIGUSR1) contador = 0;
    else if (s == SIGUSR2) {
        char buf[80];
        sprintf(buf, "Valor contador: %d\n", contador);
        write(1, buf, strlen(buf));
    }
}



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

    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigaddset(&mask, SIGUSR1);
    sigaddset(&mask, SIGUSR2);
  sigprocmask(SIG_BLOCK, &mask, NULL);


    struct sigaction sa;
    sa.sa_handler = &trata;
    sa.sa_flags = SA_RESTART;
    sigfillset(&sa.sa_mask);


    sigaction(SIGALRM, &sa, NULL);
    sigaction(SIGUSR1, &sa, NULL);
    sigaction(SIGUSR2, &sa, NULL);
    while (1) {
        alarm(1);
        sigfillset(&mask);
        sigdelset(&mask, SIGALRM);
        sigdelset(&mask, SIGUSR1);
        sigdelset(&mask, SIGUSR2);
        sigdelset(&mask, SIGINT);
        sigsuspend(&mask);
    }
    return 0;
}
```

▼ eventos2.c

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

```
int contador = 0;


void trata(int s)
{
    if (s == SIGALRM) contador += 1;
    if (s == SIGUSR1) contador = 0;
    else if (s == SIGUSR2) {
        char buf[80];
        sprintf(buf, "Valor contador: %d\n", contador);
        write(1, buf, strlen(buf));
    }

    struct sigaction sa1;
    sa1.sa_handler = SIG_DFL;
    sa1.sa_flags = SA_RESETHAND;
    sigfillset(&sa1.sa_mask);

    sigaction(SIGALRM, &sa1, NULL);
    sigaction(SIGUSR1, &sa1, NULL);
    sigaction(SIGUSR2, &sa1, NULL);

}



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

    sigemptyset(&mask);
    sigaddset(&mask, SIGALRM);
    sigaddset(&mask, SIGUSR1);
    sigaddset(&mask, SIGUSR2);
  sigprocmask(SIG_BLOCK, &mask, NULL);


    struct sigaction sa;
    sa.sa_handler = &trata;
    sa.sa_flags = SA_RESTART;
    sigfillset(&sa.sa_mask);


    sigaction(SIGALRM, &sa, NULL);
    sigaction(SIGUSR1, &sa, NULL);
    sigaction(SIGUSR2, &sa, NULL);
    while (1) {
        alarm(1);
        sigfillset(&mask);
        sigdelset(&mask, SIGALRM);
        sigdelset(&mask, SIGUSR1);
        sigdelset(&mask, SIGUSR2);
        sigdelset(&mask, SIGINT);
        sigsuspend(&mask);
    }
    return 0;
}
```

▼ signal_perdido2.c

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

void error_y_exit(char *msg,int exit_status)
{
    perror(msg);
    exit(exit_status);
```

```
}

void trat_sigusr1(int s) {
    char buf[80];

    sprintf(buf, "Hijo: SIGUSR1 recibido \n");
    write(1, buf,strlen(buf));
}

void trat_sigalrm(int s) {
    char buf[80];

    sprintf(buf, "Padre: voy a mandar SIGUSR1 \n");
    write(1, buf,strlen(buf));
}

main(int argc, char *argv[]) {
    int i,pid_h;
    char buf [80];
    int delay;
    struct sigaction sa, sa2;
    sigset_t mask;

    if (argc !=2) {
  sprintf(buf, "Usage: %s delayParent \n delayParent: 0|1\n",argv[0]);
  write(2,buf,strlen(buf));
  exit(1);
    }

    delay = atoi(argv[1]);
    //signal (SIGUSR1, trat_sigusr1);
    sa.sa_handler = &trat_sigusr1;
    sa.sa_flags = SA_RESTART;
    sigfillset(&sa.sa_mask);
    if (sigaction(SIGUSR1, &sa, NULL) < 0) error_y_exit("sigaction", 1);

    //signal (SIGALRM, trat_sigalrm);
    sa2.sa_handler = &trat_sigalrm;
    sa2.sa_flags = SA_RESTART;
    sigfillset(&sa2.sa_mask);
    if (sigaction(SIGALRM, &sa2, NULL) < 0) error_y_exit("sigaction", 1);

    pid_h = fork ();

    if (pid_h == 0) {
  sigset_t delay_mask;
  sigemptyset(&delay_mask);
  sigaddset(&delay_mask, SIGUSR1);
  sigprocmask(SIG_BLOCK, &delay_mask, NULL);

  sprintf(buf, "Hijo entrando al pause\n");
  write(1,buf,strlen(buf));
  //pause();

  sigfillset(&mask);
        sigdelset(&mask, SIGUSR1);
        sigdelset(&mask, SIGINT);
        sigsuspend(&mask);

  sigprocmask(SIG_UNBLOCK, &delay_mask, NULL);
  sprintf(buf, "Hijo sale del pause\n");
  write(1,buf,strlen(buf));
    } else {
  if (delay) {
    alarm(5);
    //pause();
    sigfillset(&mask);
            sigdelset(&mask, SIGALRM);
          sigdelset(&mask, SIGINT);
          sigsuspend(&mask);
  }
  sprintf(buf, "Padre manda signal SIGUSR1\n");
  write(1,buf,strlen(buf));
```

```
  if (kill (pid_h, SIGUSR1) < 0) error_y_exit("kill", 1);
  waitpid(-1, NULL, 0);
  sprintf(buf, "Padre sale del waitpid\n");
  write(1,buf,strlen(buf));
    }
}
```

Manual

All the info on Signals.