Operációs rendszerek Bsc

3 gyak

Miskolc 2021

Készitette:

Butella Bence Kristóf

NK:IVLJQO

Feladatok

1. Készítsen egy parent.c és a child.c programokat. A parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek
szöveget ír a szabványos kimenetre (5-ször) (pl. a hallgató neve és a neptunkód)!
Mentés: parent.c, ill. child.c

Parent.c:

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

main (int argc, char *argv[],char *env[])
{
      int status, pid, ppid, gpid;

      ppid = getpid();        /* Szulo pid-jet elkaptuk */

      if ((pid = fork()) == 0)
      {     /* innen a gyermek processz */
            printf("\n A gyermek pid = %d \n", getpid());
            status = execl(strcat(env[0]+5,"/cs/child"),0);
            if (status == -1)
            {     perror("parent execl failed");
                  exit ();
            }
      }

      /* innen a szulo process */

      printf("\n %d.sz. szulo var a %d.sz. gyermekre.\n ",
ppid,pid);
      if (pid != wait(&status)) perror(" Szulo - varakozas hiba
");
}
```

Child.c:

```
#include <stdio.h>

main()
{
      int i;

      for (i=0; i< 5; i++)
      printf("\n IVLJQO.");

}
```

2. Adott a következő terhelés esetén egy rendszer.

	P1	P2	P3	P4
Érkezés	0	8	12	20
CPU idő	15	7	26	10
Indulás	0	15	22	48
Befejezés				
Várakozás				

A tanult ütemezési algoritmus (FCFS, SJF, RR: 10 ms) felhasználásával határozza meg
a.) Várakozási/átlagos várakozási időt, befejezési időt?
b.) Ábrázolja Gantt diagrammal az aktív/várakozó folyamatok futásának sorrendjét (használjon Excel or Word etc.)!

	A	B	C	D	E	F
1						
2		p1	p2	p3	p4	
3	Érkezési id	0	8	12	20	
4	CPU	15	7	10	10	
5	Indulás id	0	15	22	48	
6	Befejezés	15	22	32	58	
7	Várakozás	0	7	10	28	
8	Átlag vár	11,25				
9						
10						

3. Értelmezzék a mintapéldákat és oldják meg: alarm.c.; alarm_ado.c;

alarmra_var.c - a jegyzet 68. oldalán található.

Mentés: neptunkod_alarm.c.; neptunkod_alarm_ado.c;

neptunkod_alarmra_var.c

alarm.c:

```
/************************************************************
 * Project: Szoftverrendszerek, signal kezeles
 * File name: $HOME/cs/alarm.c
 * Abstract: Pelda signal kezelesre.
 *           ctrl/break signal hatasara a do_int handler
mukodik
 *           eloszor, majd ignoralodik ez a signal.
 *           A ciklusban masodpercenkent alarm generalodik,
 *           az alarm signal handler-e a do_nothing.
 *           A pause felfuggeszti a process-t, amig kap egy
signalt.
 * Usage:    cc -o alarm alarm.c -lc_s
 *
 *           alarm
 * Athor: Dr. Vadasz Denes
 * Internals: signal (), alarm(), pause()
 *
 * Bugs, problems: none
 * History:
 *   Created: 1993. oct. 7
 *   Modified:
************************************************************/

#include <unistd.h>
#include <signal.h>
#define SECOND 1

void do_nothing();
void do_int();

main ()
{
      int i;
      unsigned sec=1;

      signal(SIGINT, do_int);

for (i=1;i<8;i++) {
      alarm(sec);
      signal(SIGALRM, do_nothing);
      printf("  %d varok de meddig?\n",i);
      pause();
    }
}
void do_nothing(){ ;}

void do_int() {
      printf(" int jott ");
      signal(SIGINT,SIG_IGN);
}
```

alarm_ado.c:

```
/************************************************************
 * Project: Szoftverrendszerek, signal kezeles
 * File name: $HOME/cs/alarm_ado.c
 * Abstract: Pelda signal kezelesre.

 * Usage:    >cc -o alarm_ado alarm_ado.c -lc_s
 *         >#  pid = az alarmra_var pid-je
 *           >alarm_ado pid
 * Athor: Dr. Vadasz Denes
 * Internals: atoi(), perror(), exit(), kill()
 *
 * Bugs, problems: none
 * History:
 *   Created: 1993. oct. 11
 *   Modified:
************************************************************/


#include <sys/types.h>
#include <signal.h>

main(int argc, char **argv)
{
      int pid;

      if (argc < 1)
      {
            perror(" Nincs kinek");
            exit(1);
      }

      pid = atoi(argv[1]);

      kill(pid, SIGALRM);
}
```

alarmra_var.c :

```
/************************************************************
 * Project: Szoftverrendszerek, signal kezeles
 * File name: $HOME/cs/alarmra_var.c
 * Abstract: Pelda signal kezelesre.
 *           Az alarm signal handler-e a do_nothing.
 * Usage:    >cc -o alarmra_var alarmra_var.c-lc_s
 *
 *           >alarmra_var &
 *            # es jegyezd meg a pid-jet
 * Athor: Dr. Vadasz Denes
 * Internals: signal (), pause()
 *
 * Bugs, problems: none
 * History:
```

```
 *   Created: 1993. oct. 11
 *   Modified:
**************************************************************/

#include <unistd.h>
#include <signal.h>


void do_nothing();


main ()
{


      signal(SIGALRM, do_nothing);
      printf("  %d varok de meddig?\n");
      pause();
      printf("  Vegre, itt az alarm \n");
}
void do_nothing(){ ;}
```

4. a) Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetéket, a
gyerek processz beleír egy szöveget a csővezetékbe (A kiírt szöveg: XY neptunkod), a szülő
processz ezt kiolvassa, és kiírja a standard kimenetre.
Mentés: neptunkod_unnamed.c

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

int main (int argc, char *argv[])
{
    pid_t pid;
    int pipomfd[2];
    int ret;
    char buf [20];

    ret = pipe(pipomfd);

    if(ret == -1)
    {
        perror("pipe");
        exit(1);
    }

    pid=fork();

    if(pid == 0)
    {
        /* Gyerek processz*/
        close(pipomfd[0]);
        write(pipomfd[1],"IVLJQO", 7);
    }
    else
```

```
    /* SzĂĽlĂ¶ processz*/
    close(pipomfd[1]);
    read(pipomfd[0],buf,15);
    printf("buf:%s\n",buf);
}
```

b) Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített
csővezetéket (neve: neptunkod), a gyerek processz beleír egy szöveget a csővezetékbe (A
hallgató neve: pl. Keserű Ottó), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.
Mentés: neptunkod_named.c

```
// C program implementálodik FIFOvá
//Ez az Oldal ir majd olvas
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
      int fd;

      // FIFO elérési utvonal
      char * myfifo = "/tmp/myfifo";

      // Nevesitett (FIFO) készités
      // mkfifo(<pathname>, <permission>)
      mkfifo(myfifo, 0666);

      char arr1[80], arr2[80];
      while (1)
      {
            // Megnyitja csak irásra
            fd = open(myfifo, O_WRONLY);

            // Itt iratunk bele.
            // 80 elem maximum
            fgets(Butella Bence, 80, stdin);

            // Bele Irjuk és bezárjuk
            write(fd, arr2, strlen(arr2)+1);
            close(fd);

            // Meg nyitjuk a fifot csak olvasásra
            fd = open(myfifo, O_RDONLY);

            // Olvasás a fifobol
            read(fd, arr1, sizeof(arr1));

            // Kiiratjuk amit bele irtunk
            printf("User2: %s\n", arr1);
            close(fd);
```

```
        }
        return 0;
}
```