Vježba 1: Prekidi i signali

UNIX ne dozvoljava pojedinom korisniku izravno korištenje prekida procesora. Stoga prekide treba simulirati koristeći *signale* koje jezgra operacijskog sustava šalje procesima.

Zadatak

Napisati dva programa:

- `obrada` - program koji omogućava obradu prekida s više razina/prioriteta (simulira ponašanje sustava opisanog u 3. poglavlju i to bez sklopa za prihvat prekida)
- `generator_prekida` - program koji generira signale i šalje ih prvom procesu

Svaki program pokrenuti u svojoj ljusci.

Struktura prekidne rutine dana je sljedećim pseudokodom:

```
prekidna_rutina  /* pokreće se pojavom signala uz zabranu daljih prekida */
{
   odredi uzrok prekida, tj. indeks i;
   OZNAKA_ČEKANJA[i] = 1;

   ponavljaj {
      /* odredi signal najveceg prioriteta koji ceka na obradu */
      x = 0;
      za j = TEKUćI_PRIORITET + 1 do N radi {
         ako je (OZNAKA_ČEKANJA[j]<>0) tada {
            x = j;
         }
      }

      /* ako postoji signal koji čeka i prioritetniji je od trenutnog posla, idi u obradu */
      ako je (x > 0) tada {
         OZNAKA_ČEKANJA[x] = 0;
         PRIORITET[x] = TEKUĆI_PRIORITET;
         TEKUćI_PRIORITET = x;
         omogući prekidanje;
         obrada_prekida(x);
         zabrani prekidanje;
         TEKUćI_PRIORITET = PRIORITET[x];
      }

   } dok je (x > 0);
}
```

UPUTA

Sklopovski prekid u jednoj razini simulira se slanjem određene vrste signala. Nakon toga će se prekinuti izvođenje programa, zabraniti dalje prekidanje, i pozvati funkcija za obradu signala koja simulira prekidnu rutinu.

Osim izravno preko tipkovnice, proces može dobiti signal do drugog procesa (funkcija `kill(pid,sig)`). Također, iz komandne linije se naredbom `kill` može poslati signal nekom od procesa (npr. `kill -SIGINT 12345`).

Na početku prekidne rutine prvo treba zabraniti daljnje prekidanje. Kako se koristi više signala, na osnovu signala koji je uzrokovao prekid treba odrediti razinu prioriteta prekida *i*. Prekidna rutina mora biti funkcija s jednim cjelobrojnim parametrom i mora biti najavljena u glavnom programu za svaki signal koji se prihvaća naredbom:

```
sigset (SIGNAL, prekidna_rutina);
```

Program treba moći prihvatiti i obraditi pet različitih signala (odabrati iz skupa signala) s time da prekid SIGINT treba biti najveće razine. Razine prioriteta se kreću od 1 do 5 s time da veći broj označava veći prioritet.

Zabrana prekida signalom *SIGNAL* simulira se naredbom `sighold(SIGNAL)`, a omogućavanje prekida simulira se naredbom: `sigrelse(SIGNAL);`

Budući da se ovdje ista funkcija koristi za obradu više prekida funkcije `sighold` i `sigrelse` treba pozvati za svaki od signala (više uzastopnih poziva)! Zbog toga je zgodno te pozive grupirati u dvije funkcije: zabrani_prekidanje() i dozvoli_prekidanje(), kao u kosturu rješenja.

Obrada prekida ne mora ništa korisno raditi, već samo treba trajati neko vrijeme. Umjesto dugih petlji, može poslužiti petlja s naredbom *sleep(1)*.

Kostur rješenja programa za obradu prekida dan je sljedećim kodom:

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

#define N 6    /* broj razina proriteta */

int OZNAKA_CEKANJA[N];
int PRIORITET[N];
int TEKUCI_PRIORITET;


int sig[]={SIGUSR1, SIGUSR2, ..., SIGINT};
void zabrani_prekidanje(){
   int i;
   for(i=0; i<5; i++)
      sighold(sig[i]);
}
void dozvoli_prekidanje(){
   int i;
   for(i=0; i<5; i++)
      sigrelse(sig[i]);
}

void obrada_signala(int i){
   /* obrada se simulira trošenjem vremena,
      obrada traje 5 sekundi, ispis treba biti svake sekunde */
}
```

```
void prekidna_rutina(int sig){
   int n=-1;
   zabrani_prekidanje();
   switch(sig){
      case SIGUSR1:
         n=1;
         printf("- X - - - -\n");
         break;
      case SIGUSR2:
         n=2;
         printf("- - X - - -\n");
         break;
      ...
   }
   OZNAKA_CEKANJA[n]=1;
   ponavljaj{
      ...
      dozvoli_prekidanje();
      obrada_prekida(n);
      zabrani_prekidanje();
      ...
   }
   dozvoli_prekidanje();
}

int main ( void )
{
   sigset (SIGUSR1, prekidna_rutina);
   ...
   sigset (SIGINT, prekidna_rutina);

   printf("Proces obrade prekida, PID=%ld\n", getpid());
   /* troši vrijeme da se ima što prekinuti - 10 s */

   printf ("Zavrsio osnovni program\n");

   return 0;
}
```

Drugi program nasumično generira signale (četri odabrana, SIGINT nije u tom skupu) i šalje ih u slučajnim vremenskim intervalima (od 3 do 5 sekundi) procesu za obradu prekida (PID mu se zadaje preko komandne linije). Kada proces koji šalje signale, (generator) primi (npr. od korisnika preko tipkovnice) signal SIGINT, tada proces treba poslati signal SIGKILL procesu koji prihvaća signale te nakon toga i sam završiti s radom.

Kostur rješenja programa za slanje signala dan je sljedećim kodom:

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

int pid=0;

void prekidna_rutina(int sig){
   /* pošalji SIGKILL procesu 'pid'*/
   exit(0);
}

int main(int argc, char *argv[]){
   pid=atoi(argv[1]);
   sigset(SIGINT, prekidna_rutina);

   while(1){
      /* odspavaj 3-5 sekundi */
      /* slučajno odaberi jedan signal (od 4) */
      /* pošalji odabrani signal procesu 'pid' funkcijom kill*/
   }
   return 0;
}
```

Zgodno je programe pokretati iz dvije ljuske (iz jedne program za obradu prekida, a iz druge program za generiranje prekida). Program za generiranje signala mora se pokrenuti drugi da bi mogao dobiti identifikacijski broj prvog procesa kome šalje signale.

Ispis programa za obradu prekida treba izgledati vrlo slično kao u nastavku:

```
~/os1/vj1> ./obrada
Proces obrade prekida, PID=10962
G 1 2 3 4 5
-----------
1 - - - - -
2 - - - - -
3 - - - - -
4 - - - - -
```

Nakon pokretanja prvog programa i kada se zna PID prvog procesa (u ovom slučaju 10962), u drugoj ljusci (dvije ljuske možete dobiti tako da se dvaput logirate) se pokrene drugi program koji generira prekide:

```
~/os1/vj1> ./generator_prekida 10962
```

Tada bi se u prvoj ljusci trebalo ispisivati (stanja varijabli nije nužno potrebno ispisivati!):

```
GP S1 S2 S3 S4 S5
-----------------
 1  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 2  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 3  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 4  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 5  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 6  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 7  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 8  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
```

```
 9  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 -  X  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
 -  P  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  1  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  2  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  3  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  4  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  -  -  X  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  -  -  P  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  1  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  2  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  3  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  X  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  4  -  -      O_CEK[0 0 0 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  5  -  -      O_CEK[0 0 0 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  K  -  -      O_CEK[0 0 0 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  P  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  -  X  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  -  P  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  1  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  2  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  3  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  4  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  X  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  5  -      O_CEK[0 0 0 0 1 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  K  -      O_CEK[0 0 0 0 1 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  P  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  1  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  2  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  X  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  3  -      O_CEK[0 0 0 1 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  4  -      O_CEK[0 0 0 1 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  X  -  -  -      O_CEK[0 0 0 1 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  5  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  -  K  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=4  PRIOR[0 0 0 1 3 0 ]
 -  -  -  1  -  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  2  -  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  3  -  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  4  -  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
^C -  -  -  -  -  X      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  -  -  P      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  X  -  -      O_CEK[0 0 1 1 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  -  -  1      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  -  -  2      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  -  -  3      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  -  -  4      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  X  -  -      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  -  -  5      O_CEK[0 0 1 3 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  -  -  K      O_CEK[0 0 1 3 0 0 ]  TEK_PRIOR=5  PRIOR[0 0 0 1 0 3 ]
 -  -  -  5  -  -      O_CEK[0 0 1 3 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  K  -  -      O_CEK[0 0 1 3 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  P  -  -      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  1  -  -      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  2  -  -      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  X  -  -  -      O_CEK[0 0 1 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  3  -  -      O_CEK[0 0 2 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  4  -  -      O_CEK[0 0 2 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  5  -  -      O_CEK[0 0 2 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  K  -  -      O_CEK[0 0 2 2 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  P  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  1  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  2  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  3  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  4  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  5  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
```

```
 -  -  -  K  -  -      O_CEK[0 0 2 1 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  P  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  1  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  2  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  3  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  4  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  5  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  -  K  -  -      O_CEK[0 0 2 0 0 0 ]  TEK_PRIOR=3  PRIOR[0 0 0 1 0 0 ]
 -  -  P  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  1  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  2  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  3  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  4  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  5  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  K  -  -  -      O_CEK[0 0 1 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  P  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  1  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  2  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  3  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  4  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  5  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  -  K  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=2  PRIOR[0 0 1 0 0 0 ]
 -  5  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
 -  K  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=1  PRIOR[0 0 0 0 0 0 ]
10  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
11  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
12  -  -  -  -  -      O_CEK[0 0 0 0 0 0 ]  TEK_PRIOR=0  PRIOR[0 0 0 0 0 0 ]
Killed
```

X - označava pojavu signala, P početak obrade, K završetak obrade, a brojevi koliko je sekundi potrošeno na obradu.

U ispisu se mogu uočiti tri stvari. Prvo, `^C` znači da je procesu došao signal SIGINT s tipkovnice, a drugo `Killed` znači da je program za slanje signala završio s radom (i prije toga poslao SIGKILL). I treće, u primjeru ispisa program pamti više prekida iste razine (OZNAKA_ČEKANJA[i] se ne postavlja u 1 kao što je navedeno u pseudokodu, već se povećava za 1).