Operativni sistemi

Signali i redovi poruka

(Laboratorijska vežba III-1)

Zadatak 1

Korišćenjem programskog jezika C napisati UNIX/Linux program koji onemogućava da se tekući proces prekine jednim pritiskom kombinacije tastera Ctrl-C. Proces se prekida tek kada se ova kombinacija tastera pritisne dva puta za redom.

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

/* Funkcija za obradu signala */
void catch_int(int sig_num)
{
  /* Postavlja se default funkcija za obradu signala za sledeci put kada bude
   *pritisnuta kombinacija Ctrl-C */
  signal(SIGINT, SIG_DFL);
  printf("Nije moguce prekinuti program!\n");
  fflush(stdout);
}

int main(int argc, char* argv[])
{
  /* Postavlja funkciju catch_int da obradjuje signal SIGINT */
  signal(SIGINT, catch_int);
  /* Beskonacna petlja */
  for ( ;; )
    pause();
}
```

Zadatak 2

Korišćenjem programskog jezika C napisati UNIX/Linux program koji može da se prekine samo ako se kombinacija tastera Ctrl-C pritisne pet puta za redom. Ukoliko korisnik pritisne kombinaciju tastera Ctrl-Z program na ekranu ispisuje koliko puta do sada je pritisnuto Ctrl-C.

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

/* Brojac Ctrl-C */
int ctrl_c_count = 0;
#define CTRL_C_THRESHOLD 5

/* Funkcija za obradu Ctrl-C signala */
void catch_int(int sig_num)
{
  /* Ponovo postavlja signal handler */
  signal(SIGINT, catch_int);
  /* Inkrementira brojac i proverava da li je pritisnuto Ctrl-C dovoljan broj
   *puta */
```

```
  ctrl_c_count++;

  if (ctrl_c_count >= CTRL_C_THRESHOLD)
  {
    char answer[30];
    /* Pita korisnika da li stvarno zeli da prekine proces */
    printf("\nRealy Exit? [y/N]: ");
    fflush(stdout);
    fgets(answer, sizeof(answer), stdin);
    if (answer[0] == 'y' || answer[0] == 'Y') {
      printf("\nExiting...\n");
      fflush(stdout);
      exit(0);
    }
    else
    {
      printf("\nContinuing\n");
      fflush(stdout);
      /* reset Ctrl-C counter */
      ctrl_c_count = 0;
    }
  }
}

/* Ctrl-Z signal handler */
void catch_suspend(int sig_num)
{
  /* Ponovo postavlja signal handler */
  signal(SIGTSTP, catch_suspend);
  /* Stampa Ctrl-C brojac */
  printf("\n\nDo sada, Ctrl-C je pritisnuto '%d' puta\n\n", ctrl_c_count);
  fflush(stdout);
}

int main(int argc, char* argv[])
{
  /* Postavlja Ctrl-C and Ctrl-Z signal handlere */
  signal(SIGINT, catch_int);
  signal(SIGTSTP, catch_suspend);

  /* Beskonacna petlja */
  for ( ;; )
    pause();

  return 0;
}
```

Zadatak 3

Korišćenjem programskog jezika C napisati UNIX/Linux program koji od korisnika očekuje da sa tastature unese korisničko ime. Ukoliko korisnik ne unese ništa u roku od 30 sekundi proces obaveštava korisnika da je vreme za unos isteklo i izlazi.

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

```
char user[40];

/* Handler za alarm */
void catch_alarm(int sig_num)
{
  printf("Operacija je istekla. Izlazim...\n\n");
  exit(0);
}
int main(int argc, char* argv[])
{
  /* Signal handler za ALRM signal */
  signal(SIGALRM, catch_alarm);
  /* Prompt korisniku da unese string */
  printf("Username: ");
  fflush(stdout);
  /* Startuje alarm od 30 sekundi */
  alarm(30);
  /* Ceka da korisnik unese ime */
  gets(user);
  /* Ukoliko korisnik unese ime uklanja alarm */
  alarm(0);
  printf("Username: '%s'\n", user);
  return 0;
}
```

Zadatak 4

Korišćenjem programskog jezika C napisati UNIX/Linux program koji svom procesu detetu, korišćenjem redova poruka, prosleđuje ulaz koji prima preko tastature, a process dete dobijene poruke štampa na ekranu. Predvideti da se unosom teksta “QUIT” prekida rad programa.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

#define MSGLEN 50

struct mymsgbuf
{
  long mtype;
  char mtext[MSGLEN];
};

main()
{
  int msqid;
  struct mymsgbuf buf;

  buf.mtype = 1;
  strcpy(buf.mtext, "");

  msqid = msgget(10104, 0666 | IPC_CREAT);

  if (fork() == 0)
```

```
  {
    while(strcmp(buf.mtext, "quit") != 0)
    {
      msgrcv(msqid, &buf, 50, 0/*sve poruke*/, 0/*blokirajuci */);
      printf("Proces dete: %s\n", buf.mtext);
    }
  }
  else
  {
    while(strcmp(buf.mtext, "quit") != 0)
    {
      scanf("%s", buf.mtext);
      msgsnd(msqid, &buf, strlen(buf.mtext)+1, 0/*blokirajuci */);
    }

    wait(NULL);
    msgctl(msqid, IPC_RMID, NULL);
  }

}
```

Zadatak 5

Korišćenjem programskog jezika C napisati UNIX/Linux program koji simulira problem proizvođač/potrošač korišćenjem redova poruka (message-queues). Glavni program se deli u dva procesa. Prvi proces (proizvođač) kreira N slučajnih pozitivnih celih brojeva i šalje ih drugom procesu. N se određuje tokom izvršenja, takođe kao slučajan pozitivan ceo broj. Po završetku slanja, prvi proces šalje -1 kao kod za kraj. Drugi proces (potrošač) preuzima poslate brojeve iz poruka i štampa ih na standardnom izlazu.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define RED_PORUKA 10001
#define MAX_PORUKA 10

struct poruka
{
  long tip;
  char tekst[MAX_PORUKA];
};

int main()
{
  int pid;
  int redid;
  int broj = 0;
  struct poruka bafer;

  //Kreiramo novi proces
  pid = fork();

  if (pid < 0)
```

```
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
    exit(1);
  }
  if (pid == 0)
  {
    //Izvrsava pbroj);roces dete
    if (execl("zadatak5b", "zadatak5b", NULL) < 0)
    {
      printf("Doslo je do greske prilikom  ucitavanja izvrsne datoteke\n");
    }

    exit(1);
  }

  //Kod koji izvrsava samo proces roditelj
  //Pribavlja se referenca na red poruka. Red se kreira ukoliko ne postoji
  redid = msgget(RED_PORUKA, IPC_CREAT | 0666);
  if (redid < 0)
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
    exit(1);
  }
  //primaju se poruke iz reda poruka sve dok ne stigne poruka -1
  do
  {
    if (msgrcv(redid, &bafer, MAX_PORUKA, 0, 0) < -1)
    {
      printf("Doslo je do greske prilikom prijema poruke\n");
      exit(1);
    }
    broj = atoi(bafer.tekst);
    printf("%d\n", broj);
  }
  while( broj > -1);

  wait(NULL);
  msgctl(redid, IPC_RMID, 0);
  return 0;
}



#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define RED_PORUKA 10001
#define MAX_PORUKA 10

struct poruka
{
  long tip;
  char tekst[MAX_PORUKA];
};
int main()
```

```
{
  int N; //broj poruka koje ce biti poslate
  int redid;
  int i;
  struct poruka bafer;

  //Pribavlja se referenca na red poruka. Red se kreira ukoliko ne postoji
  redid = msgget(RED_PORUKA, IPC_CREAT | 0666);
  if (redid < 0)
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
    exit(1);
  }

  //Odredjuej se broj poruka koje ce biti poslate
  N = rand() % 20;
  //U red se salje N slucajnih pozitivnih celih brojeva + (-1) na kraju za kraj
komunikacije
  for (i = 0; i < N + 1; i++)
  {
    if (i == N)
    {
      sprintf(bafer.tekst, "%d", -1);//poslednja poruka je -1
    }
    else
    {
      sprintf(bafer.tekst, "%d", rand() % 100);//slucajno generisani brojevi
    }
    //Tip poruke u ovom primeru nije bitan i postavlja se na 1
    bafer.tip = 1;
    //Poruka se salje u red
    if (msgsnd(redid, &bafer, sizeof(bafer.tekst), 0) < -1)
    {
      printf("Doslo je do greske prilikom prijema poruke\n");
      exit(1);

    }
  }
  return 0;
}
```

Zadatak 6

Korišćenjem programskog jezika C napisati UNIX/Linux programkoji učitava podatke iz tekstualne datoteke cela (red po red) i zatim korišćenjem reda poruka sve parne redove šalje procesu koji konvertuje sva slova u velika i zapisuje ih u datoteku pola1, a sve neparne redove procesu koji konvertuje sva slova u mala i zapisuje ih u datoteku pola2.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

#define RED_PORUKA 10301
#define MAX_PORUKA 255
```

```
struct poruka
{
  long tip;
  char tekst[MAX_PORUKA];
};

int main()
{
  int pid;
  int redid;
  struct poruka bafer;
  FILE * f;
  int poruka;
  char tmp[MAX_PORUKA];

  //Pribavlja se referenca na red poruka. Red se kreira ukoliko ne postoji
  redid = msgget(RED_PORUKA, IPC_CREAT | 0666);

  //Kreiramo proces za parne linije
  pid = fork();
  if (pid < 0)
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
    exit(1);
  }

  if (pid == 0)
  {
    //Izvrsava proces dete
    if (execl("zadatak6a", "zadatak6a", "pola1", "1",  NULL) < 0)
    {
      printf("Doslo je do greske prilikom ucitavanja izvrsnedatoteke\n");
    }

    exit(1);
  }

  //Kreiramo proces za neparne linije
  pid = fork();
  if (pid < 0)
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
    exit(1);
  }
  if (pid == 0)
  {
    //Izvrsava proces dete
    if (execl("zadatak6a", "zadatak6a", "pola2", "2",  NULL) < 0)
    {
      printf("Doslo je do greske prilikom ucitavanja izvrsnedatoteke\n");
    }

    exit(1);
  }

  //Kod koji izvrsava samo proces roditelj
  if (redid < 0)
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
```

```
    exit(1);
  }

  f = fopen("poruke", "r");
  if (f == NULL)
  {
    printf("Doslo je do greske prilikom otvaranja datoteke\n");
    exit(1);
  }

  //Cita se linija po linija iz datoteke
  poruka = 0;
  fgets(tmp, MAX_PORUKA, f);
  do
  {
    strcpy(bafer.tekst, tmp);

    if (poruka % 2 == 0)
      bafer.tip = 1;//parne poruke su tipa 1
    else
      bafer.tip = 2;//neparne poruke su tipa 2

    if (msgsnd(redid, &bafer, MAX_PORUKA, 0) < 0)
    {
      printf("Doslo je do greske prilikom slanja poruke.\n");
      exit(1);
    }

    poruka++;
    fgets(tmp, MAX_PORUKA, f);

  }
  while(!feof(f));

  fclose(f);
  //saljemo poruku sa sadrzajem END za oba procesa
  strcpy(bafer.tekst, "END");
  bafer.tip = 1;
  msgsnd(redid, &bafer, MAX_PORUKA, 0);
  bafer.tip = 2;
  msgsnd(redid, &bafer, MAX PORUKA, 0);

  //Red poruka se brise tek kada se zavrse oba deteta
  wait(NULL);
  wait(NULL);
  msgctl(redid, IPC_RMID, 0);
  return 0;
}

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define RED_PORUKA 10301
#define MAX_PORUKA 255
struct poruka
```

```
{
  long tip;
  char tekst[MAX_PORUKA];
};

int main(int argc, char* argv[])
{
  int redid;
  struct poruka bafer;
  FILE * f;
  int tip;

  if (argc < 3)
  {
    printf("Nema dovoljno ulaznih argumenata za pokretanje procesa\n");
    exit(1);
  }
  //Pribavlja se referenca na red poruka. Red se kreira ukoliko ne postoji
  redid = msgget(RED_PORUKA, IPC_CREAT | 0666);
  if (redid < 0)
  {
    printf("Doslo je do greske prilikom kreiranja novog procesa\n");
    exit(1);
  }

  tip = atoi(argv[2]);
  if (tip == 0)
  {
    printf("Pogresan tip poruka\n");
    exit(1);
  }

  f = fopen(argv[1], "w");
  if (f == NULL)
  {
    printf("Doslo je do greske prilikom otvaranja datoteke\n");
    exit(1);
  }

  //Cita poruke iz reda
  while (strcmp(bafer.tekst, "END") != 0)
  {
    if (msgrcv(redid, &bafer, MAX_PORUKA, tip, 0) < 0)//Citaju se samo poruke
odredjenog tipa
    {
      printf("Doslo je do greske prilikom prijema poruke.\n");
      break;
    }

    if (strcmp(bafer.tekst, "END") != 0)
    {
      fprintf(f, "%s\n", bafer.tekst);
      fflush(f);
    }
  }

  fclose(f);
  return 0;
}
```