Operativni sistema

Linux System programming i Procfs datotečni sistem

(Laboratorijska vežba I-3)

Zadatak 1

a) Korišćenjem programskog jezika C napisati Linux program koji sa komandne linije prihvata određeni broj argumenata (realnih brojeva). Program treba na ekranu da odštampa broj prosleđenih argumenata, da sračuna sumu i srednju vrednost niza argumenata i dobijene vrednosti odštampa na ekranu.

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

main(int argc, char* argv[])
{
        double sum, avg, tmp;
        int i;

        printf("Broj argumenata komandne linije: %d\n", argc);

        sum = 0.0;
        avg = 0.0;

        for(i=1 ; i<argc ; i++){
                tmp = atof(argv[i]);
                printf("Argument %d: %s %f\n", i, argv[i], tmp);
                sum += atof(argv[i]);
                printf("Parcijalna suma u iteraciji %d: %f\n", i,
sum);
        }
        avg = sum / (double)(argc-1);
        printf("Suma argumenata komandne linije: %f\n", sum);
        printf("Srednja vrednost argumenata je: %f\n", avg);

return 0;
}
```

b) Prevesti kreirani program korišćenjem gcc prevodioca. **(gcc program.c, gcc program.c –o program)**

c) Izvršiti datoteku dobijenu prevođenjem programa u prethodnom koraku. **(./a.out, ./program)**

d) Kompajlirati kreirani program tako da izvršna datoteka može da se debagira. Uočite razliku u veličini izvršne datoteke kada se prevođenje vrši sa informacijama za debagiranje i bez njih. **(gcc program.c –g –o program, datoteka za debagiranje je znatno veca zato što sadrži informacije nophodne za debagiranje)**

e) Startovati debugger gdb nad izvršnom datotekom dobijenom u prethodnom koraku. **(gdb program)**

f) Postavit break point na liniju u kojoj je promenljiva za sumiranje realnih brojeva inicijalizuje na vrednost 0. **(break 11)**

g) Pokrenuti izvršenje programa a kada se on zaustavi na breakpoint-u pogledati vrednost promenljive za sumiranje relnih brojeve. **(run, print semid, step, next)**
h) U liniji gde se vrši sumiranje realnih brojeva, postaviti uslovni breakpoint da bi se utvrdila vrednost sume kada se dodaje realni broj koji je veći od 30.0. **(delete 1, break 18, condition 2 sum > 30)**
i) Korišćenjem komande help pogledati koje sve opcije postoje za rad sa breakpointima. (**help**, **help running**, **help advance**)
j) Izaći iz debuggera. (**quit**)
k) Modifikovati program iz tačke a) tako da se realni brojevi unose sa tastature sve dok korisnik ne unese ključnu reč KRAJ.

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

main()
{
        double sum, avg, tmp;
        int i;
        char input[10];

        sum = 0.0;
        avg = 0.0;
        i = 0;

        printf("Unesite realan broj ili KRAJ za kraj rada\n");
        scanf("%s", input);
        i++;

        while (strcmp(input, "KRAJ") != 0){
                tmp = atof(input);
                printf("Argument %d: %s %f\n", i, input, tmp);
                sum += atof(input);
                printf("Parcijalna suma u iteraciji %d: %f\n", i,
sum);

                scanf("%s", input);
                i++;
        }
        avg = sum / (double)(i-1);
        printf("Suma argumenata komandne linije: %f\n", sum);
        printf("Srednja vrednost argumenata je: %f\n", avg);

        return 0;
}
```

Zadatak 2

Korišćenjem programskog jezika C napisati Linux program koji koji korišćenjem procfs interfejsa određuje i na standardnom izlazu štampa informacije:

a) PID-u i PPID-u tekućeg procesa (programa koji ste kreirali i koji pristupa procfs-u).

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

#define MAX_BUF 100

main(int argc, char * argv[])
{
   char buf[MAX_BUF], buf1[MAX_BUF], buf2[MAX_BUF];
   FILE  * f;
   int i;

   for (i = 0; i < MAX_BUF; i++)
   {
         buf[i] = '\0';
         buf1[i] = '\0';
         buf2[i] = '\0';
   }

   f = fopen("/proc/self/status", "r");

   if (f == 0)
   {
         printf("Doslo je do greske prilikom pristupanja informacijama\n");
         return -1;
   }

   while (!feof(f))
   {
         fscanf(f, "%s", buf);

         if (strcmp(buf, "Pid:") == 0)
         {
               fscanf(f, "%s", buf1);
               printf("Pid je: %s\n", buf1);
         }

         if (strcmp(buf, "PPid:") == 0)
         {
               fscanf(f, "%s", buf2);
               printf("PPid je: %s\n", buf2);
         }

   }

   fclose(f);

   return 0;

}
```

b) Statusu i mapi zauzeću memorijskih regiona za proces čiji je PID = 1.

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

#define MAX_BUF 500

main(int argc, char * argv[])
{
   char buf[MAX_BUF];
   FILE  * f;
   int i;

   for (i = 0; i < MAX_BUF; i++)
         buf[i] = '\0';

   f = fopen("/proc/1/status", "r");

   if (f == 0)
   {
         printf("Doslo je do greske prilikom pristupanja informacijama\n");
         return -1;
   }

   while (!feof(f))
   {
         if (fgets(buf,MAX_BUF, f))
               printf("%s", buf);
   }

   fclose(f);

   f = fopen("/proc/1/maps", "r");

   if (f == 0)
   {
         printf("Doslo je do greske prilikom pristupanja informacijama\n");
         return -1;
   }

   while(!feof(f))
   {
         if (fgets(buf,MAX_BUF, f))
               printf("%s", buf);
   }
   fclose(f);

   return 0;

}
```

c) Svim particijama kojje postoje u sistemu (njihova imena i veličina u megabajtima).

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

#define BUF_SIZE 100

int main()
{

    char buf[BUF_SIZE];
    char name[BUF_SIZE];
    char size[BUF_SIZE];
    FILE * f;
    int i, j;

    f = fopen("/proc/partitions", "r");

    if (!f)
    {
        printf("Doslo   je   do   greske   prilikom   pristupanja   datoteci   sa
informacijama o memoriji\n");
        return -1;
    }

    j = 0;
    while (!feof(f))
    {
        if (j > 1)
   {
         for (i = 0; i < 4; i++)
         {
               if (i == 2)
                     fscanf(f, "%s", size);
               else if (i == 3)
                     fscanf(f, "%s", name);
               else
                     fscanf(f, "%s", buf);
         }

         if (!feof(f))
               printf("%s\t%s\n", name, size);
   }
   else
         fgets(buf, BUF_SIZE, f);
   j++;
    }

    fclose(f);

    return 0;
}
```

d) Svim blok uređajima koji su prisutni u sistemu.

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

#define BUF_SIZE 100

int main()
{
    char buf[BUF_SIZE];
    int i;
    FILE * f;
    int print_flag = 0;

    for (i = 0; i < BUF_SIZE; i++)
        buf[i] = '\0';

    f = fopen("/proc/devices", "r");

    if (!f)
    {
        printf("Doslo je do greske prilikom otvaranja datoteke sa
informacijama\n");
        return -1;
    }

    while (!feof(f))
    {
        if (fgets(buf, BUF_SIZE, f))
        {

            if (strstr(buf, "Block devices:"))
              print_flag = 1;

             if (print_flag == 1)
               printf("%s", buf);
        }
    }

    fclose(f);

    return 0;

}
```