Технически университет - София

Направление - ФКСТ

СИСТЕМНО ПРОГРАМИРАНЕ

ДОКУМЕНТАЦИЯ НА КУРСОВА РАБОТА

Студент: *Денислав Петков, 29 група*

Специалност: *Компютърно и софтуерно инженерство*

Факултетен номер – *121219031*

Ръководител: *Елена Антонова*

Съдържание

Анализ на изготвеното приложение

Условие:

Във файл са записани няколко IP адреса. Стартират се два процеса, като единият прочита адресите от файла и ги подава на втория процес. Вторият процес създава и стартира дъщерни процеси, които ping-ват дадените адреси. Ако няма връзка с дадения адрес, то той се изпраща обратно към първият процес и се извежда IP-то със съобщение за грешка.

Приложението при пускане трябва да стартира 2 процеса, единият трябва да чете адресите, които трябва да бъдат „пингнати“ от файл и да ги предава към 2-рия процес. А 2-рият процес за всеки адрес стартира дъщерен процес, на който му предава един адрес, който той да „пингне“. Ако резултатен е неуспешен, то този адрес се връща директно към 1-вия родителски процес, който го изкарва на екрана със съобщение за грешка.

Функционално описание на приложението

При стартирането на приложение се създават 2 тръби(pipes) и 2 процеса.

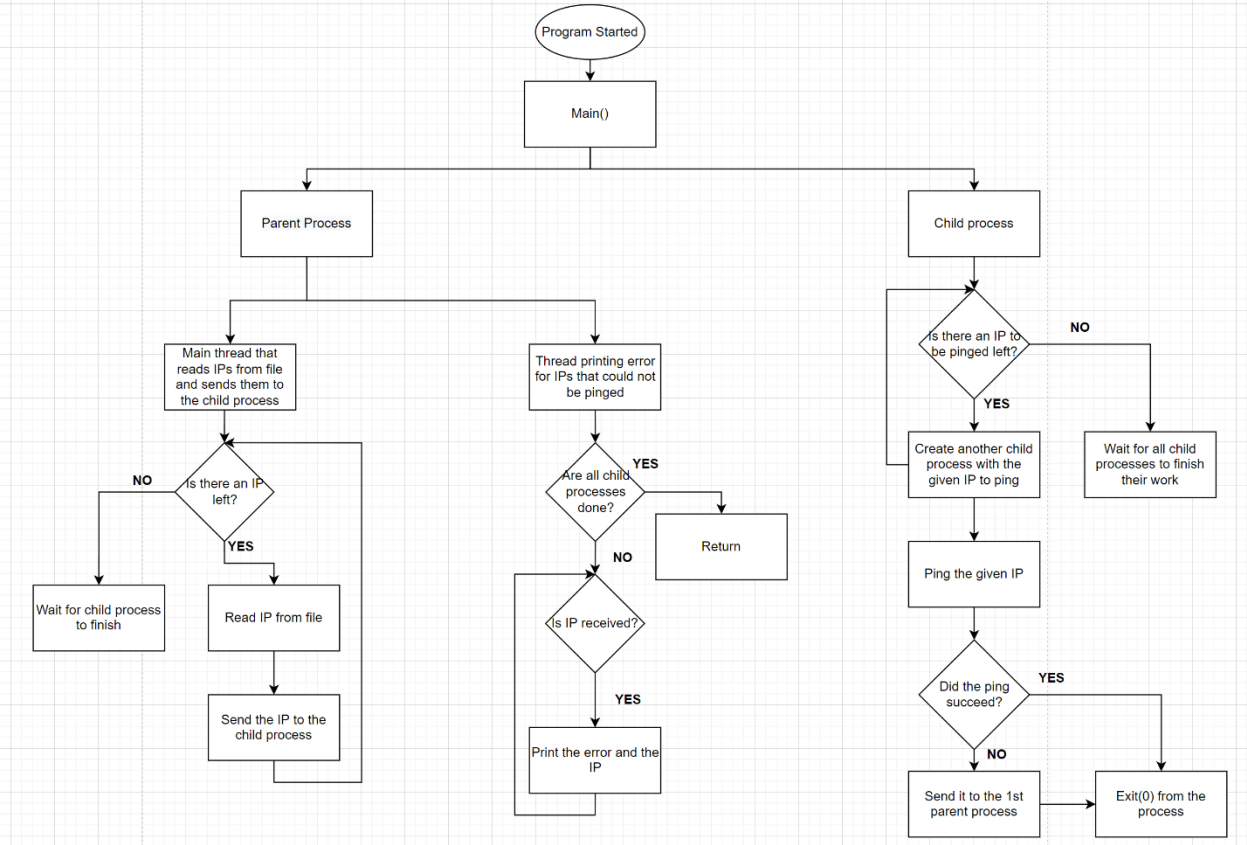
Едната тръба служи за комуникация между родителския процес и дъщерния. А другата за комуникация между дъщерните процеси на дъщерния и 1-вият родителски процес.

В родителския процес се извършва четенето на адресите от файла. Те се четат ред по ред от файла и се изпращат чрез тръба на 1-рия дъщерен процес. В същото време се стартира и 2-ра нишка, която следи за идващи адреси, които не са успели да се пингнат.

В дъщерния процес се създават нови дъщерни процеси, като на всеки му се подава по един адрес, който да пингне. При неуспех, този адрес се изпраща чрез тръба към първият родителски процес.

Програмата приключва когато всички адреси са вече пингнати.

Workflow Diagram:



Изпълнение на функционалностите

Добавени са 4 функции:

```
void parentProcess();
void *printErrorThread();
void childProcess();
void startChild(char *ip);
```

В „parentProcess()“ е логиката, която се изпълнява в родителския процес. Тоест стартирането на нишката за изкарването на екрана на адресите, които не могат да бъдат пингнати, четенето от файла и предаването на адресите към дъщерния процес.

```
void parentProcess()
{
    // close reading end
    close(pipeParentToChild[0]);

    // close writing end
    close(pipeChildrenToParent[1]);

    FILE *ipsFile;

    ipsFile = fopen("ips.dat", "r");
    if (ipsFile == NULL)
    {
        perror("Failed");
        exit(1);
    }

    int currentIP_len = 256;
    char currentIP[currentIP_len];

    pthread_t tid;
    pthread_create(&tid, NULL, &printErrorThread, NULL);
    pthread_detach(tid);
```

```
    // send the ips to the child
    while (!feof(ipsFile))
    {
        fgets(currentIP, currentIP_len, ipsFile);
        write(pipeParentToChild[1], currentIP, strlen(currentIP) + 1);
        sleep(2);
    }

    fclose(ipsFile);

    // close writing end
    close(pipeParentToChild[1]);

    wait(NULL);
}
```

„printErrorThread()“ е функцията за нишката, която изкарва на екрана адреса, заедно със съобщение за грешка, ако той не е успешно пингнат.

```
void *printErrorThread()
{
    char returnedIP[256];

    while (read(pipeChildrenToParent[0], returnedIP, sizeof(returnedIP)) > 0)
    {
        printf("Could not reach %s\n", returnedIP);
    }

    // close reading end
    close(pipeChildrenToParent[0]);

    return NULL;
}
```

В „childProcess()“ е логиката на първият дъщерен процес. Той чете адресите от родителския процес и стартира нов дъщерен процес за всеки прочетен адрес.

```
void childProcess()
{
    // close writing end
    close(pipeParentToChild[1]);

    // close reading end
    close(pipeChildrenToParent[0]);

    char ip[128];

    while (read(pipeParentToChild[0], ip, sizeof(ip)) > 0)
    {
        startChild(ip);
    }

    // close reading end
    close(pipeParentToChild[0]);

    wait(NULL);

    // close writing end
    close(pipeChildrenToParent[1]);
}
```

„startChild(char *ip)“ приема като аргумент адрес, който трябва да пингне. Вътре в него става пингването на адреса. Ако то не е успешно изпраща адресът към първият родителски процес.

```
void startChild(char *ip)
{
    FILE *fp;

    char commandOutputLine[1024];
    char commandOutput[8192];

    if (fork() == 0)
    {
        // redirect the stderr to stdout with timeout set to 1 sec
        char command[64] = "/bin/ping 2>&1 -c 1 -w 1 ";
        strcat(command, ip);
```

```
        fp = popen(command, "r");

        if (fp == NULL)
        {
            printf("Failed to run command\n");
            exit(1);
        }

        while (fgets(commandOutputLine, sizeof(commandOutputLine), fp) != NULL)
        {
            strcat(commandOutput, commandOutputLine);
        }

        if (strstr(commandOutput, "Name or service not known") != NULL ||
strstr(commandOutput, "100% packet loss") != NULL)
        {
            write(pipeChildrenToParent[1], ip, strlen(ip) + 1);
        }
        pclose(fp);
        exit(0);
    }
}
```

Експериментални данни

Приложението е тествано със следните адреси:

- google.com
- test.com
- facebook.com
- youtube.com
- hehe.bg
- psTestAddress.com
- cnn.com
- random.org
- istio.io
- shouldNotWork.bg
- lastOne.com

Един адрес се приема за неуспешно пингнат при следните грешки:

- 1 packets transmitted, 0 received, 100% packet loss, time 0ms
- Name or service not known

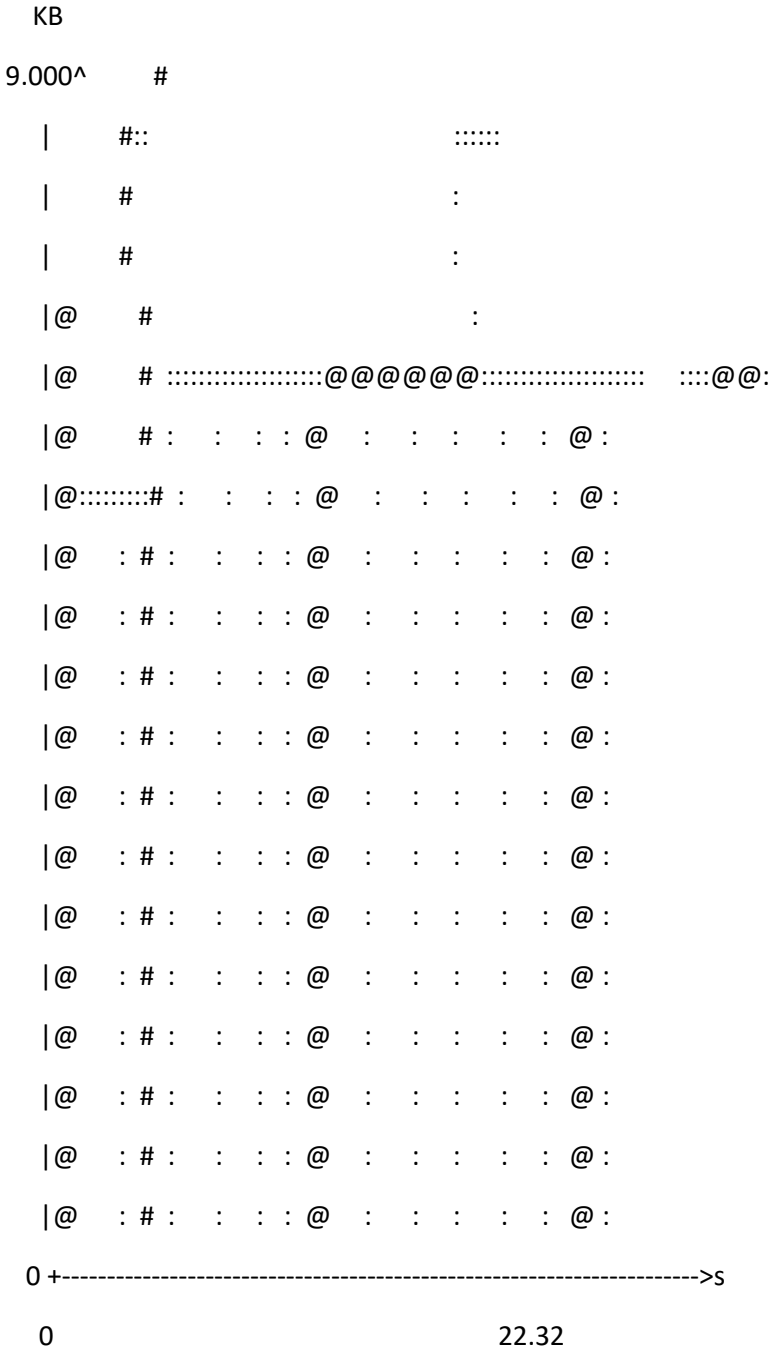
Тоест случаите, когато пингът не може да достигне до дадения адрес, или този адрес не може да бъде намерен се считат за неуспешни.

Резултат при изпълнението на програмата с тези адреси:

```
√ /mnt/c/Users/denis/Desktop/workspace/denislavPetkov/TU/SP(main) % ./a.out
Could not reach test.com

Could not reach hehe.bg

Could not reach psTestAddress.com

Could not reach shouldNotWork.bg

Could not reach lastOne.com
```

Адресите са пингнави 1 по 1 в терминал и съвпадат с резултатите получени след изпълнението на програмата.

Анализ на използваната памет от приложението



Number of snapshots: 51

Detailed snapshots: [4, 14, 24 (peak), 32, 42]

```
--------------------------------------------------------------------------------
 n      time(ms)       total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 0          0            0            0          0         0
 1          4          288            0          0       288
 2          8        1,152            0          0     1,152
 3         12        6,432            0          0     6,432
 4         16        7,392            0          0     7,392
00.00% (0B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.

--------------------------------------------------------------------------------
 n      time(ms)       total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 5         20        3,984            0          0     3,984
 6         23        4,584            0          0     4,584
 7         47        4,576            0          0     4,576
 8         48        5,464            0          0     5,464
 9        266        5,456            0          0     5,456
10        270        1,160            0          0     1,160
11        274        1,688            0          0     1,688
12        278          880            0          0       880
13        282           64            0          0        64
14        286          192            0          0       192
00.00% (0B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.

--------------------------------------------------------------------------------
 n      time(ms)       total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
15        288          224            0          0       224
```

16	290	968	472	16	480
17	292	1,056	472	16	568
18	294	1,336	472	16	848
19	296	1,392	744	24	624
20	299	6,032	4,840	32	1,160
21	2,299	6,192	4,840	32	1,320
22	2,300	6,016	4,840	32	1,144
23	3,329	6,192	4,840	32	1,320
24	3,331	9,216	5,864	40	3,312

63.63% (5,864B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.

->55.56% (5,120B) 0x48EDD33: _IO_file_doallocate (filedoalloc.c:101)

| ->55.56% (5,120B) 0x48FDEFF: _IO_doallocbuf (genops.c:347)

| ->44.44% (4,096B) 0x48FCCD3: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:486)

| | ->44.44% (4,096B) 0x48FDFB5: _IO_default_uflow (genops.c:362)

| | ->44.44% (4,096B) 0x48EF89B: _IO_getline_info (iogetline.c:60)

| | ->44.44% (4,096B) 0x48EE6F9: fgets (iofgets.c:53)

| | ->44.44% (4,096B) 0x1095D1: parentProcess (program.c:65)

| | ->44.44% (4,096B) 0x109467: main (program.c:32)

| |

| ->11.11% (1,024B) 0x48FCF5F: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)

| ->11.11% (1,024B) 0x48FB6E4: _IO_new_file_xsputn (fileops.c:1244)

| ->11.11% (1,024B) 0x48FB6E4: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1197)

| ->11.11% (1,024B) 0x48E29A1: __vfprintf_internal (vfprintf-internal.c:1373)

| ->11.11% (1,024B) 0x48CDD6E: printf (printf.c:33)

| ->11.11% (1,024B) 0x109694: printErrorThread (program.c:84)

| ->11.11% (1,024B) 0x4851608: start_thread (pthread_create.c:477)

| ->11.11% (1,024B) 0x498B162: clone (clone.S:95)

|

->05.12% (472B) 0x48EE95D: __fopen_internal (iofopen.c:65)

| ->05.12% (472B) 0x48EE95D: fopen@@GLIBC_2.2.5 (iofopen.c:86)

| ->05.12% (472B) 0x1094C5: parentProcess (program.c:48)

| ->05.12% (472B) 0x109467: main (program.c:32)

|

->02.95% (272B) 0x40149CA: allocate_dtv (dl-tls.c:286)

  ->02.95% (272B) 0x40149CA: _dl_allocate_tls (dl-tls.c:532)

    ->02.95% (272B) 0x4852322: allocate_stack (allocatestack.c:622)

      ->02.95% (272B) 0x4852322: pthread_create@@GLIBC_2.2.5 (pthread_create.c:660)

        ->02.95% (272B) 0x1095AE: parentProcess (program.c:59)

          ->02.95% (272B) 0x109467: main (program.c:32)

----

 n   time(ms)   total(B)   useful-heap(B) extra-heap(B)   stacks(B)

----

 25   3,333   8,888   5,864   40   2,984

 26   3,335   8,808   5,864   40   2,904

 27   4,300   7,224   5,864   40   1,320

 28   6,300   7,224   5,864   40   1,320

 29   8,301   7,224   5,864   40   1,320

 30   9,363   7,224   5,864   40   1,320

 31   10,301   7,224   5,864   40   1,320

 32   10,400   7,224   5,864   40   1,320

81.17% (5,864B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.

->70.87% (5,120B) 0x48EDD33: _IO_file_doallocate (filedoalloc.c:101)

| ->70.87% (5,120B) 0x48FDEFF: _IO_doallocbuf (genops.c:347)

| ->56.70% (4,096B) 0x48FCCD3: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:486)

| | ->56.70% (4,096B) 0x48FDFB5: _IO_default_uflow (genops.c:362)

| | ->56.70% (4,096B) 0x48EF89B: _IO_getline_info (iogetline.c:60)

| | ->56.70% (4,096B) 0x48EE6F9: fgets (iofgets.c:53)

```
|   |      ->56.70% (4,096B) 0x1095D1: parentProcess (program.c:65)
|   |        ->56.70% (4,096B) 0x109467: main (program.c:32)
|   |
|   ->14.17% (1,024B) 0x48FCF5F: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)
|     ->14.17% (1,024B) 0x48FB6E4: _IO_new_file_xsputn (fileops.c:1244)
|       ->14.17% (1,024B) 0x48FB6E4: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1197)
|         ->14.17% (1,024B) 0x48E29A1: __vfprintf_internal (vfprintf-internal.c:1373)
|           ->14.17% (1,024B) 0x48CDD6E: printf (printf.c:33)
|             ->14.17% (1,024B) 0x109694: printErrorThread (program.c:84)
|               ->14.17% (1,024B) 0x4851608: start_thread (pthread_create.c:477)
|                 ->14.17% (1,024B) 0x498B162: clone (clone.S:95)
|
->06.53% (472B) 0x48EE95D: __fopen_internal (iofopen.c:65)
| ->06.53% (472B) 0x48EE95D: fopen@@GLIBC_2.2.5 (iofopen.c:86)
|   ->06.53% (472B) 0x1094C5: parentProcess (program.c:48)
|     ->06.53% (472B) 0x109467: main (program.c:32)
|
->03.77% (272B) 0x40149CA: allocate_dtv (dl-tls.c:286)
  ->03.77% (272B) 0x40149CA: _dl_allocate_tls (dl-tls.c:532)
    ->03.77% (272B) 0x4852322: allocate_stack (allocatestack.c:622)
      ->03.77% (272B) 0x4852322: pthread_create@@GLIBC_2.2.5 (pthread_create.c:660)
        ->03.77% (272B) 0x1095AE: parentProcess (program.c:59)
          ->03.77% (272B) 0x109467: main (program.c:32)


--------------------------------------------------------------------------------
  n      time(ms)       total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 33       12,301          7,224          5,864          40       1,320
 34       14,301          7,224          5,864          40       1,320
```

35	16,301	7,224	5,864	40	1,320
36	16,302	7,080	5,864	40	1,176
37	18,302	7,224	5,864	40	1,320
38	18,342	7,224	5,864	40	1,320
39	18,343	9,048	5,864	40	3,144
40	20,302	7,224	5,864	40	1,320
41	20,303	7,256	5,864	40	1,352
42	21,498	7,224	5,864	40	1,320

81.17% (5,864B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.

->70.87% (5,120B) 0x48EDD33: _IO_file_doallocate (filedoalloc.c:101)

| ->70.87% (5,120B) 0x48FDEFF: _IO_doallocbuf (genops.c:347)

| ->56.70% (4,096B) 0x48FCCD3: _IO_file_underflow@@GLIBC_2.2.5 (fileops.c:486)

| | ->56.70% (4,096B) 0x48FDFB5: _IO_default_uflow (genops.c:362)

| | ->56.70% (4,096B) 0x48EF89B: _IO_getline_info (iogetline.c:60)

| | ->56.70% (4,096B) 0x48EE6F9: fgets (iofgets.c:53)

| | ->56.70% (4,096B) 0x1095D1: parentProcess (program.c:65)

| | ->56.70% (4,096B) 0x109467: main (program.c:32)

| |

| ->14.17% (1,024B) 0x48FCF5F: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)

| ->14.17% (1,024B) 0x48FB6E4: _IO_new_file_xsputn (fileops.c:1244)

| ->14.17% (1,024B) 0x48FB6E4: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1197)

| ->14.17% (1,024B) 0x48E29A1: __vfprintf_internal (vfprintf-internal.c:1373)

| ->14.17% (1,024B) 0x48CDD6E: printf (printf.c:33)

| ->14.17% (1,024B) 0x109694: printErrorThread (program.c:84)

| ->14.17% (1,024B) 0x4851608: start_thread (pthread_create.c:477)

| ->14.17% (1,024B) 0x498B162: clone (clone.S:95)

|

->06.53% (472B) 0x48EE95D: __fopen_internal (iofopen.c:65)

| ->06.53% (472B) 0x48EE95D: fopen@@GLIBC_2.2.5 (iofopen.c:86)

```
|   ->06.53% (472B) 0x1094C5: parentProcess (program.c:48)
|     ->06.53% (472B) 0x109467: main (program.c:32)
|
->03.77% (272B) 0x40149CA: allocate_dtv (dl-tls.c:286)
  ->03.77% (272B) 0x40149CA: _dl_allocate_tls (dl-tls.c:532)
    ->03.77% (272B) 0x4852322: allocate_stack (allocatestack.c:622)
      ->03.77% (272B) 0x4852322: pthread_create@@GLIBC_2.2.5 (pthread_create.c:660)
        ->03.77% (272B) 0x1095AE: parentProcess (program.c:59)
          ->03.77% (272B) 0x109467: main (program.c:32)
```

n	time(ms)	total(B)	useful-heap(B)	extra-heap(B)	stacks(B)
43	22,303	7,224	5,864	40	1,320
44	22,304	6,752	5,864	40	848
45	22,314	6,744	5,864	40	840
46	22,316	2,624	1,768	32	824
47	22,318	2,040	1,296	16	728
48	22,320	2,016	1,296	16	704
49	22,323	944	272	8	664
50	22,324	680	0	0	680

Може да се види, че при стартирането на приложението паметта използвана от процес 1 е 0 и в продължение на времето започва да се дига. Когато достигне своят максимум, не спада веднага а поддържа горе-долу еднакъв разход до края на приложението.

Което е и очакваното поведение, понеже преди краят на приложението бива освободена използваната от приложението памет.

Приложение

Проектът може да бъде намерен в github на следния линк – github.com/denislavPetkov/SP.