

## Операционные системы и сети

### Лабораторная работа №4 Джугели Дмитрий ПИН-31Д

## Процессы в ОС Linux (II)

**Цель работы:** знакомство с системными вызовами для синхронизации процессов и обработки сигналов; изучение распределения виртуальной памяти процесса.

### Задание 1.

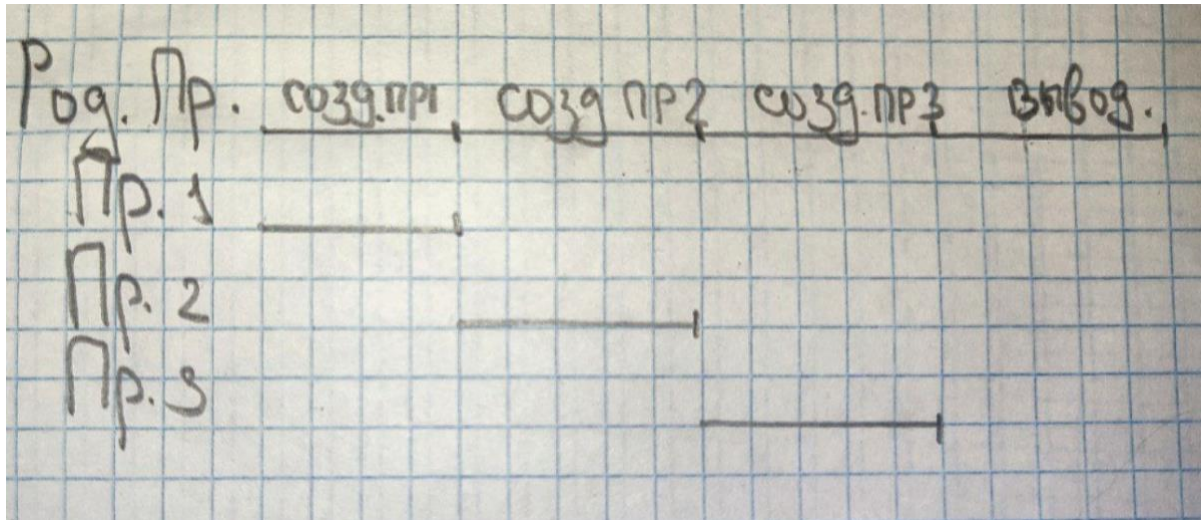
**1.1.** В программе **pr1.c** основной процесс создает три дочерних процесса и ожидает их завершения. *Выполните программу и запишите выходные данные. Нарисуйте приблизительную временную диаграмму работы всех четырех процессов.*

```
/* pr1.c */
#include <stdio.h>
#include <unistd.h>
int main()
{int i, pid, status, w
for (i=0; i<3; ++i)
    {
        pid = fork();
        if (pid ==0)
            exit(getpid() % 256);
    }
while ((w = wait(&status))&& w!= -1)
    printf ("Child  %x  returns status  %x\n", w, status);
return 0;
}
```

```

1 error generated.
ant_daddy@Dmitriy 123 % gcc pr4_1.c -o Pr4_1
ant_daddy@Dmitriy 123 % ./Pr4_1
Child acd5 returns status d500
Child acd6 returns status d600
Child acd7 returns status d700
ant_daddy@Dmitriy 123 %

```



**1.2.** Замените `exit` на `kill(pid_дочернего_процесса, SIGKILL)`. Поясните в отчете полученные в 1.1, 1.2 коды возврата дочернего процесса (см. лаб. работу 3).

```

zsh: no such file or directory: ./Pr4_2
ant_daddy@Dmitriy 123 % ./Pr4_2
Child af53 returns status 5300
Child af54 returns status 5400
Child af55 returns status 5500
zsh: killed ./Pr4_2
ant_daddy@Dmitriy 123 %

```

В программе 1.2 создаются три дочерних процесса, каждый из которых завершается с кодом возврата 0. После ожидания завершения всех дочерних процессов родительский процесс принудительно завершает все дочерние процессы сигналом SIGKILL.

**1.3.** Выполните программу **pr2.c**, где основной процесс создает дочерний процесс, завершающийся немедленно. Затем основной процесс "засыпает" на 30 секунд. Запустите программу, и, пока она выполняется, из другого окна введите команду

```
% ps -mA
```

Выпишите и поясните информацию о состоянии обоих процессов. Повторите эту команду после завершения работы программы. Объясните результаты в отчете.

```
/* pr2.c */
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main ()
{
    pid_t child_pid;
    /* Create a child process. */
    child_pid = fork ();
    if (child_pid > 0) sleep (30);
    else exit (0);
    return 0;
}
```

Информация о состоянии обоих процессов

```
501 45155 656 4006 0 31 0 34121128 520 - S+ 0 ttys000 0:00.00 ./Pr2
```

```
501 45156 45155 2006 0 0 0 0 0 - Z+ 0 ttys000 0:00.00 <defunct>
```

Через 30 секунд процессы завершаются

```

0 45165 45085 4106 0 31 0 34122020 1116 - R+ 0 ttys001 0:00.01 ps -mAl
200 295 1 4004 0 31 0 33663228 1100 - S 0 ?? 0:04.62 /usr/sbin/distnoted agent
0 140 1 4004 0 4 0 33728900 1076 - Ss 0 ?? 0:02.91 /usr/sbin/syslogd
0 461 1 4004 0 20 0 33720192 1056 - Ss 0 ?? 0:00.05 /usr/libexec/securityd_servic
0 1191 1 4004 0 4 0 33711892 1016 - Ss 0 ?? 0:00.06 /usr/libexec/maintenance
0 163 1 4004 0 37 0 33695504 1004 - Ss 0 ?? 0:00.02 /usr/sbin/KernelEventAgent
205 280 1 4004 0 4 0 33703696 988 - S 0 ?? 0:00.06 /usr/sbin/cfprepsd agent
501 590 1 4004 0 4 0 33720072 968 - Ss 0 ?? 0:00.02 /System/Library/PrivateFramew
0 217 1 4004 0 4 0 33710860 960 - S 0 ?? 0:00.02 /System/Library/Frameworks/Co
202 264 1 4004 0 31 0 33663228 960 - S 0 ?? 0:03.91 /usr/sbin/distnoted agent
88 387 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.81 /usr/sbin/distnoted agent
247 482 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.63 /usr/sbin/distnoted agent
55 462 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.68 /usr/sbin/distnoted agent
260 707 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.62 /usr/sbin/distnoted agent
92 443 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.76 /usr/sbin/distnoted agent
277 881 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.56 /usr/sbin/distnoted agent
262 255 1 4004 0 31 0 33663228 952 - S 0 ?? 0:03.91 /usr/sbin/distnoted agent
89 362 1 4004 0 31 0 33663228 952 - S 0 ?? 0:03.03 /usr/sbin/distnoted agent
278 359 1 4004 0 31 0 33663228 952 - S 0 ?? 0:03.14 /usr/sbin/distnoted agent
242 325 1 4004 0 31 0 33663228 952 - S 0 ?? 0:03.47 /usr/sbin/distnoted agent
296 1198 1 4004 0 31 0 33663228 952 - S 0 ?? 0:02.42 /usr/sbin/distnoted agent
205 275 1 4004 0 31 0 33663228 952 - S 0 ?? 0:03.81 /usr/sbin/distnoted agent
0 946 1 4004 0 4 0 33702656 908 - Ss 0 ?? 0:00.02 /System/Library/PrivateFramew
0 222 1 4004 0 4 0 33720148 896 - Ss 0 ?? 0:00.02 /System/Library/Frameworks/Ne
501 796 1 4004 0 4 0 33710932 892 - Ss 0 ?? 0:00.02 /System/Library/Frameworks/Ne
0 243 1 4004 0 31 0 33685840 800 - Ss 0 ?? 0:00.02 /usr/libexec/multiversed
0 143 1 4004 0 37 0 33692940 756 - Ss 0 ?? 0:00.03 /usr/libexec/thermalmonitord
0 180 1 4004 0 37 0 33727976 680 - Ss 0 ?? 0:00.06 aslmanager
270 451 1 4004 0 63 0 34125272 460 - Ss 0 ?? 0:00.02 /System/Library/DriverExtensi
270 450 1 4004 0 63 0 34124248 460 - Ss 0 ?? 0:00.02 /System/Library/DriverExtensi
270 377 1 4004 0 63 0 34122164 452 - Ss 0 ?? 0:00.05 /System/Library/DriverExtensi
0 160 1 4004 0 4 0 33693912 216 - Ss 0 ?? 0:00.03 /usr/libexec/dirhelper
501 41857 484 6004 0 0 0 0 - Z 0 ?? 0:00.00 <defunct>
501 43061 484 6004 0 0 0 0 - Z 0 ?? 0:00.00 <defunct>
501 2198 484 6004 0 0 0 0 - Z 0 ?? 0:00.00 <defunct>
501 819 484 6004 0 0 0 0 - Z 0 ?? 0:00.00 <defunct>
501 44273 484 6004 0 0 0 0 - Z 0 ?? 0:00.00 <defunct>
ant_daddy@Dmitriy 123 %

```

## Задание 2.

**2.1.** Завершите **bash** сигналом **SIGTERM**, затем - сигналом **SIGKILL**. Поясните результаты и запишите Ваши команды. Определите и выпишите номера этих двух сигналов.

Номера сигналов:

- SIGTERM: 15

- SIGKILL: 9

```
123 — -zsh — 85x7
^C
[ant_daddy@Dmitriy 123 % ./Pr2
[ant_daddy@Dmitriy 123 % ./Pr2
[ant_daddy@Dmitriy 123 % ./Pr2
zsh: terminated ./Pr2
ant_daddy@Dmitriy 123 %
```

```
ant_daddy — -zsh — 80x24
Last login: Thu Jun 13 12:46:29 on ttys001
ant_daddy@Dmitriy ~ % kill -15 45231
ant_daddy@Dmitriy ~ %
```

```
123 — -zsh — 85x7
[ant_daddy@Dmitriy 123 % ./Pr2
[ant_daddy@Dmitriy 123 % ./Pr2
[ant_daddy@Dmitriy 123 % ./Pr2
zsh: terminated ./Pr2
[ant_daddy@Dmitriy 123 % ./Pr2
zsh: killed ./Pr2
ant_daddy@Dmitriy 123 %
```

```
ant_daddy — -zsh — 80x24
Last login: Thu Jun 13 12:46:29 on ttys001
[ant_daddy@Dmitriy ~ % kill -15 45231
[ant_daddy@Dmitriy ~ % kill -9 45243
ant_daddy@Dmitriy ~ %
```

При использовании SIGTERM процесс получает сигнал о том, что ему нужно завершиться, и может выполнить необходимые операции перед закрытием. В случае использования SIGKILL процесс будет немедленно прерван без возможности выполнить какие-либо действия перед завершением.

**2.2.** Напишите программу **pr3.c**, которая содержит функцию-обработчик для сигнала **SIGCHLD**, обработчик вызывает функцию **wait** и записывает код возврата дочернего процесса в некоторую глобальную переменную. Основным процесс создает дочерний процесс (который завершается немедленно и возвращает какой-нибудь код возврата, например, 2), затем основной процесс в цикле выполняет некоторую работу и проверяет значение глобальной переменной. После завершения дочернего процесса распечатывает его код возврата. Используйте следующий код для функции-обработчика сигнала и для задания реакции на сигнал **SIGCHLD**:

```

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

sig_atomic_t exstatus;

void chld_hand (int snumber)
{
    int status;
    wait (&status);
    /* Store its exit status in a global variable. */
    exstatus = status;
}

int main ()
{struct sigaction sigact;
    memset (&sigact, 0, sizeof (sigact));
    sigact.sa_handler = chld_hand;
    sigaction (SIGCHLD, &sigact, NULL);
    ...
    return 0;
}

```

ПРИМЕЧАНИЕ. Так как обработчик может быть прерван в любом месте, например, поступлением другого сигнала, то присвоение значения глобальной переменной должно быть неделимой (атомарной) операцией. Это обеспечивается использованием переменной типа **sig\_atomic\_t**.

```

pr3.c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#include <unistd.h>

int exstatus;

void chld_hand(int snumber)
{
    pid_t pid;
    int status;

    while ((pid = waitpid(-1, &status, WNOHANG)) > 0)
    {
        exstatus = status;
    }
}

int main() {
    exstatus = 0;

    struct sigaction act;
    act.sa_handler = chld_hand;
    sigemptyset(&act.sa_mask);
    act.sa_flags = SA_RESTART | SA_NOCLDSTOP;
    sigaction(SIGCHLD, &act, 0);

    pid_t pid = fork();

    if (pid == -1) {
        perror("fork");
        exit(1);
    } else if (pid == 0) {
        // дочерний процесс
        exit(2);
    }

    // родительский процесс
    while (1) {
        if (exstatus != 0) {
            printf("Child process exited with status: %d\n", WEXITSTATUS(exstatus));
            break;
        }
        sleep(1);
    }

    return 0;
}

```

```

ant_daddy@Dmitriy 123 % gcc pr3.c -o Pr3
ant_daddy@Dmitriy 123 % ./Pr3
Child process exited with status: 2
ant_daddy@Dmitriy 123 %

```

### 2.3. Реализация тайм-аута при помощи сигнала-"будильника" SIGALRM.

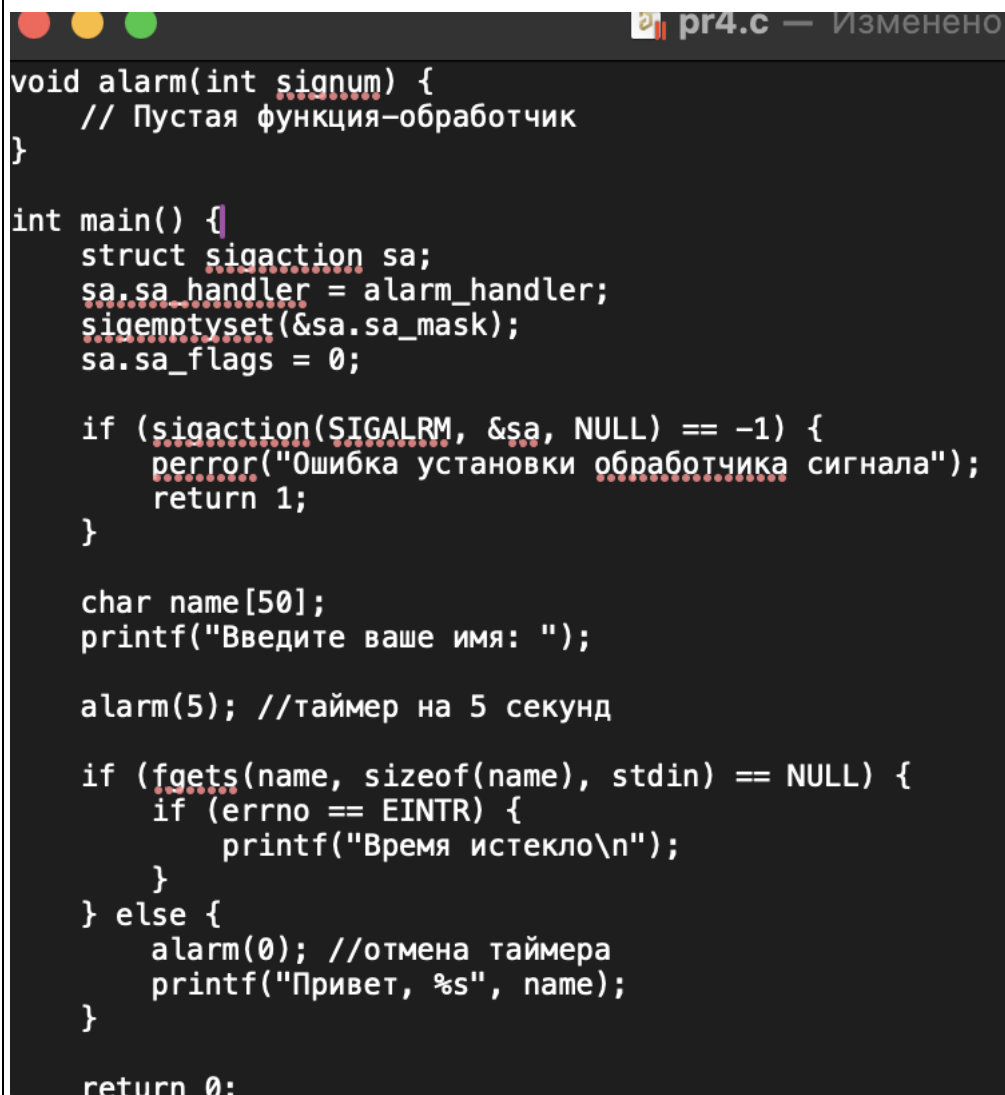
Напишите программу **pr4.c**, которая ожидает ввода имени с клавиатуры,

печатает "Привет, <имя>" и завершается. Если имя не было введено в течение пяти секунд, то программа печатает "Время истекло" и завершается.

ПРИМЕЧАНИЯ.

1. Задайте обработчик для сигнала **SIGALRM** с пустым телом функции-обработчика.
2. Перед вызовом функции **read** запустите будильник функцией **alarm**.
3. Если **read** завершилась с ошибкой, то следует проверить значение **errno**. Если **errno** равно **EINTR**, то истек тайм-аут.

Поясните в отчете, почему возникает ошибка **EINTR**. Повлияет ли на работу программы установка флага **SA\_RESTART** в поле **sa\_flags** структуры **sigaction**? Почему?



```
pr4.c — Изменено

void alarm(int signum) {
    // Пустая функция-обработчик
}

int main() {
    struct sigaction sa;
    sa.sa_handler = alarm_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;

    if (sigaction(SIGALRM, &sa, NULL) == -1) {
        perror("Ошибка установки обработчика сигнала");
        return 1;
    }

    char name[50];
    printf("Введите ваше имя: ");

    alarm(5); //таймер на 5 секунд

    if (fgets(name, sizeof(name), stdin) == NULL) {
        if (errno == EINTR) {
            printf("Время истекло\n");
        }
    } else {
        alarm(0); //отмена таймера
        printf("Привет, %s", name);
    }

    return 0;
}
```



```
[ant_daddy@Dmitriy 123 % ./Pr4
Введите ваше имя: Время истекло
[ant_daddy@Dmitriy 123 % ./Pr4
Введите ваше имя: Дмитрий
Привет, Дмитрий
```

**2.4.\* Одновременное поступление сигналов одного типа.** Напишите программу **pr5.c**.

*Основной процесс:* создает пять потомков и затем выполняет какой-либо бесконечный цикл.

*Код потомка:* **sleep(1); exit(...);**

*Код обработчика сигнала SIGCHLD:* **sleep(3); wait(0); write(1, "process terminated\n", 19);**

*Объясните в отчете полученные результаты. Сколько сигналов обработано и почему?* Модифицируйте программу, попытавшись обработать все сигналы следующими способами:

a) (**pr5a**) используя в обработчике вместо **wait** функцию **waitpid** в цикле.

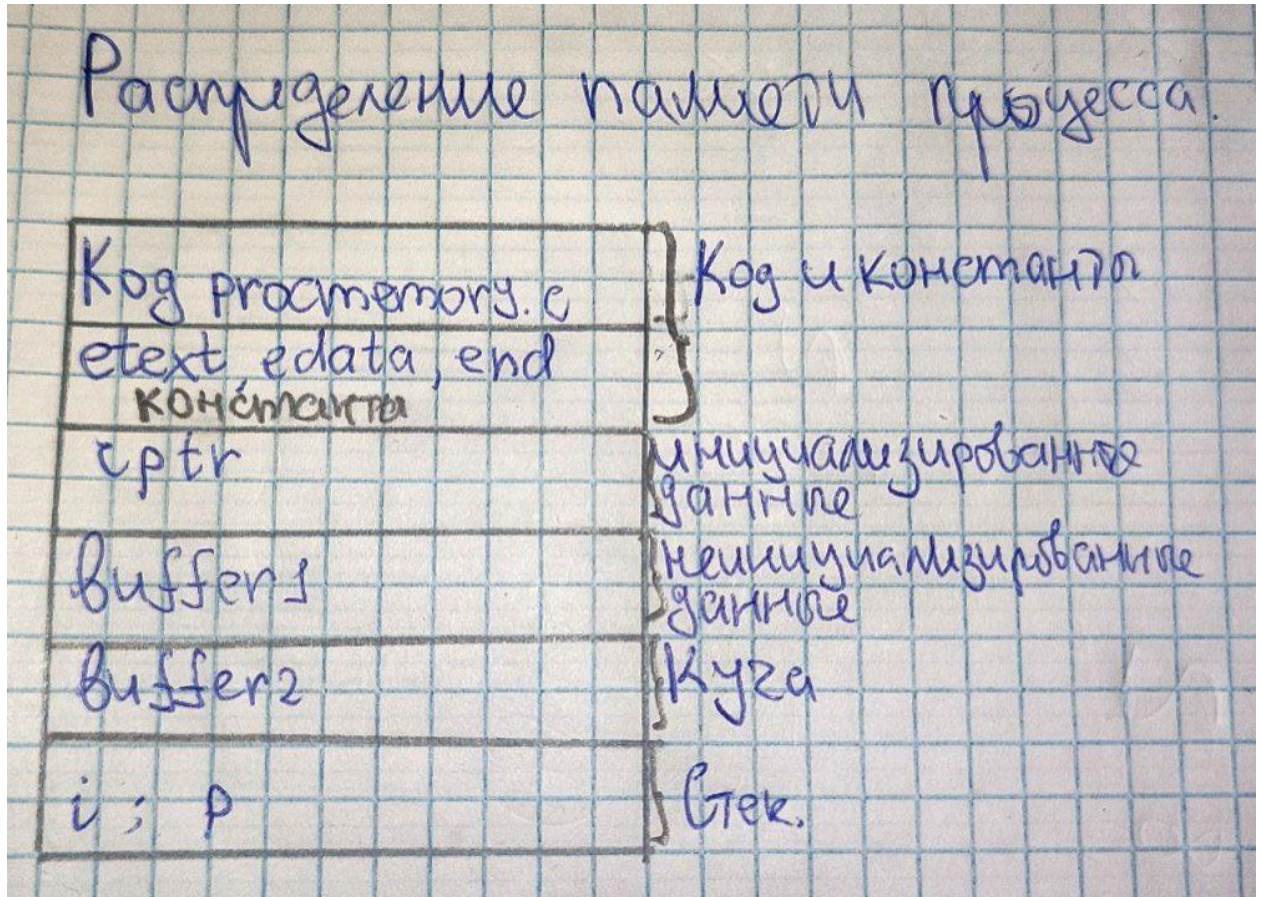
b) (**pr5b**) используя флаг **SA\_NOMASK** в поле **sa\_flags** структуры **sigaction** (см. **man**).

c) (**pr5c**) используя вместо обработчика сигнала функцию **sigwait**.

---

\* Необязательное задание. Оценивается дополнительно, если студент полностью выполнил обязательные задания.

**Задание 3. 3.1.** Используя начальную версию программы **procmemory.c** , зарисуйте в отчете распределение памяти процесса, как это сделано на рис. 1. Покажите на рисунке расположение и адреса **main**, **showit**, **cptr**, **buffer1**, **i** и память, выделенную в куче для **buffer2**.



```
ant_daddy@Dmitriy 123 % gcc procmemory.c -o PROCMEMORY
ant_daddy@Dmitriy 123 % ./PROCMEMORY
Address of main: 0x100c92e30
Address of showit: 0x100c92e20
Address of cptr: 0x7ff7bf26d690
Address of buffer1: 0x7ff7bf26d6ae
Address of buffer2: 0x7ff7bf26d6a4
Address of i: 0x7ff7bf26d69c
ant_daddy@Dmitriy 123 %
```

**3.2.** Вставьте в программу системный вызов **fork** после оператора **int i=0**. Что будет напечатано? Убедитесь, что данные (например, **i**) в обоих процессах имеют одинаковые виртуальные адреса.

После добавления **fork()** получаем:

```

ant_daddy@Dmitriy 123 % gcc procmemory.c -o PROCMEMORY
ant_daddy@Dmitriy 123 % ./PROCMEMORY
Address of main: 0x106c9ee20
Address of showit: 0x106c9ee10
Address of cptr: 0x7ff7b9261690
Address of buffer1: 0x7ff7b92616ae
Address of buffer2: 0x7ff7b92616a4
Address of i: 0x7ff7b926169c
Address of main: 0x106c9ee20
Address of showit: 0x106c9ee10
Address of cptr: 0x7ff7b9261690
Address of buffer1: 0x7ff7b92616ae
Address of buffer2: 0x7ff7b92616a4
Address of i: 0x7ff7b926169c
ant_daddy@Dmitriy 123 %

```

```

/* procmemory.c Program to display address information about the
process. Adapted from Gray, J.S., Interprocess Communication in
UNIX: The Nooks & Crannies, Prentice-Hall, 1997 */

```

```

#include <stdio.h>

```

```

#include <string.h>

```

```

#include <sys/types.h>

```

```

#include <stdlib.h>

```

```

#include <unistd.h>

```

```

/* Below is a macro definition */

```

```

#define SHW_ADR(ID,I) (printf("ID %s \t is at virt.address:
%8X\n", ID,&I))

```

```

extern int etext, edata, end; /* Global variables for process
memory */

```

```

char *cptr="This message is output by showit()\n"; /*Static */

```

```

char buffer1[25];

```

```

int showit (); /* Function prototype */

```

```

main ()
{
int i = 0;                /* Automatic variable */

/* Printing addressing information */
printf ("\nAddress etext: %8X \n", &etext);
printf ("Address edata: %8X \n", &edata);
printf ("Address end   : %8X \n", &end);
SHW_ADR ("main", main);
SHW_ADR ("showit", showit);
SHW_ADR ("cptr", cptr);
SHW_ADR ("buffer1", buffer1);
SHW_ADR ("i", i);

strcpy (buffer1, "A demonstration\n"); /* Library function */
write (1, buffer1, strlen(buffer1) + 1); /* System call */
showit(cptr);
} /* end of main function */


int showit (p)             /* A function follows */
char * p;
{
char *buffer2;
SHW_ADR("buffer2", buffer2);
if ((buffer2 = (char *)malloc ((unsigned) (strlen(p) + 1))) !=
NULL)
{
printf("Alocated memory at %X\n", buffer2);
strcpy (buffer2, p);    /* copy the string */
printf ("%s", buffer2); /* Didplay the string */
free (buffer2);        /* Release location */

```

```
    }  
else  
    {  
        printf ("Allocation error\n");  
        exit (1);  
    }  
}
```