

Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления  
Кафедра Интеллектуальных информационных технологий

ОТЧЁТ

Лабораторная работа №2

**“Создание и управление процессов в UNIX-подобных ОС”**

Выполнил:

Заяц Д. А., Готин И.

Проверил:

Цирук В. А.

Минск 2022

Цель лабораторной работы:

Научиться создавать процессы и потоки, а также управлять ими.

Общее задание:

Написать программу, создающую два дочерних процесса с использованием двух вызовов `fork()`. Родительский и два дочерних процесса должны выводить на экран свой `pid` и `pid` родительского процесса и текущее время в формате: часы : минуты : секунды : миллисекунды. Используя вызов `system()`, выполнить команду `ps -x` в родительском процессе. Найти свои процессы в списке запущенных процессов.

Листинг:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <time.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6 #include <stdlib.h>
7
8 int main(int argc, char *argv[])
9 {
10     time_t t = time(NULL);
11     struct tm *showtime = localtime(&t);
12     pid_t pid;
13     printf("Родительский процесс создан, его pid: %d\n", getpid());
14     printf("%02d:%02d:%02d\n", showtime->tm_hour, showtime->tm_min, showtime->tm_sec);
15     for(int i = 0; i < 2; i++){
16         if((pid = fork()) == 0){
17             printf("Дочерний процесс %d создан, его pid: %d, его ppid: %d\n", i+1, getpid(), getppid());
18             printf("%02d:%02d:%02d\n", showtime->tm_hour, showtime->tm_min, showtime->tm_sec);
19             exit(0);
20         }
21         system("ps -x");
22         wait(0);
23     }
24     exit(0);
25     return 0;
26 }
```

Создание первого дочернего процесса:

```
Родительский процесс создан, его pid: 10183
Дочерний процесс 1 создан, его pid: 10184, его ppid: 10183
14:55:48
```

Родительский и дочерний процессы:

```
10183 pts/0    S+      0:00  ./t1.exe
10184 pts/0    Z+      0:00  [t1.exe] <defunct>
10185 pts/0    S+      0:00  sh -c ps -x
10186 pts/0    R+      0:00  ps -x
```

Создание второго дочернего процесса:

```
10186 pts/0    R+      0:00  ps -x
Дочерний процесс 2 создан, его pid: 10187, его ppid: 10183
14:55:48
```

Родительский и дочерний процессы:

```
10183 pts/0    S+      0:00  ./t1.exe
10187 pts/0    Z+      0:00  [t1.exe] <defunct>
10188 pts/0    S+      0:00  sh -c ps -x
10189 pts/0    R+      0:00  ps -x
```

Индивидуальное задание 1:

Написать программу синхронизации двух каталогов, например, Dir1 и Dir2. Пользователь задаёт имена Dir1 и Dir2. В результате работы программы файлы, имеющиеся в Dir1, но отсутствующие в Dir2, должны скопироваться в Dir2 вместе с правами доступа. Процедуры копирования должны запускаться в отдельном процессе для каждого копируемого файла. Каждый процесс выводит на экран свой pid, имя копируемого файла и число скопированных байт. Число одновременно работающих процессов не должно превышать N (вводится пользователем).

Листинг:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4  #include <dirent.h>
5  #include <unistd.h>
6  #include <fstream>
7  #include <sys/wait.h>
8  #include <sys/stat.h>
9  #include <fcntl.h>
10 #include "iostream"
11 #include "vector"
12 #include <exception>
13 #include <filesystem>
14 #define BUF_SIZE 4096
15 char buffer [BUF_SIZE];
16
17
18 using namespace std;
19
20 void mycopyfile (const char *source_file, const char *destination_file)
21 {
22     int infd, outfd;
23     ssize_t bytes;
24
25     infd = open (source_file, O_RDONLY);
26     if (infd == -1)
27     {
28         fprintf (stderr, "НЕ могу открыть входящий файл \"(%s)\n", source_file);
29         exit (1);
30     }
31
32     outfd = open (destination_file, O_WRONLY | O_CREAT | O_TRUNC, 0640);
33     if (outfd == -1)
34     {
35         fprintf (stderr, "НЕ могу открыть выходящий файл \"(%s)\n", destination_file);
36         exit (2);
37     }
38
39     while ((bytes = read(infd, buffer, BUF_SIZE)) > 0)
40         write (outfd, buffer, bytes);
41
42     close (infd);
43     close (outfd);
44     //exit (3);
45 }
46
47
48
```

```

49
50
51 int get_file_size(std::string filename) // path to file
52 {
53     FILE *p_file = NULL;
54     p_file = fopen(filename.c_str(), "rb");
55     fseek(p_file, 0, SEEK_END);
56     int size = ftell(p_file);
57     fclose(p_file);
58     return size;
59 }
60
61
62 int main (void)
63 {
64     DIR *dir1, *dir2;
65     struct dirent *entry1, *entry2;
66     int flag, flag_name, k, i, m, cur, NPROC;
67     string in_str, out_str;
68     pid_t pid_end;
69     int file_o;
70     string str_dir1, str_dir2;
71     cout << "Dir1 = ";
72     cin >> str_dir1; // = "/Users/ignat/Desktop/dir1";
73     cout << "Dir2 = ";
74     cin >> str_dir2; // = "/Users/ignat/Desktop/dir2";
75     cout << "Число процессов N: \n"; // Введите число процессов
76     cin >> NPROC; // Число процессов по адресу NPROC
77     pid_t pids[10];
78     dir1 = opendir(str_dir1.c_str()); // Результат открытия каталога Dir1
79     dir2 = opendir(str_dir2.c_str()); // Результат открытия каталога Dir2
80
81     if ((!dir1)||(!dir2)) // Если неудачное открытие
82     {
83         perror ("diropen");
84         return 1;
85     }
86
87     i = 0;
88     vector<dirent *> files_in_dir1;
89     while ((entry1 = readdir(dir1)) != NULL) {
90         if (entry1->d_name[0] == '.')
91             continue;
92         else;
93         files_in_dir1.push_back(entry1);

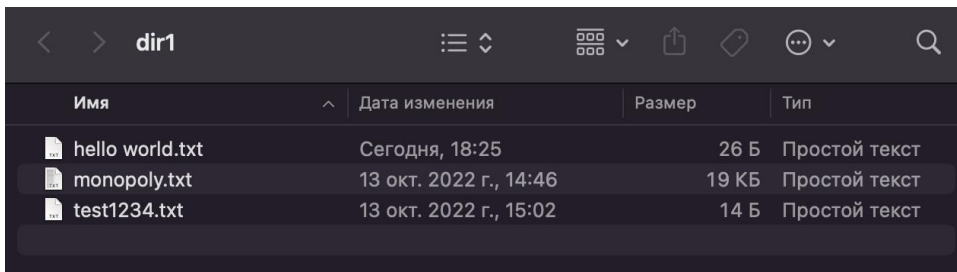
```

```

94     }
95
96     vector<dirent *> files_in_dir2;
97     while ((entry1 = readdir(dir2)) != NULL) {
98         if (entry1->d_name[0] == '.')
99             continue;
100         else;
101         files_in_dir2.push_back(entry1);
102     }
103
104     for(int n = 0; n < files_in_dir1.size(); n++) {
105         flag = 1;
106         for(int j = 0; j < files_in_dir2.size(); j++) {
107             for (flag_name = 1, k = 0; (k < MAXNAMLEN) && (files_in_dir1[n]->d_name[k] != '\0'); k++) {
108                 if (files_in_dir1[n]->d_name[k] != files_in_dir2[j]->d_name[k]) // Если кат. не равны...
109                     flag_name = 0;
110             }
111
112             if ((files_in_dir2[j]->d_type == files_in_dir1[n]->d_type) &&
113                 (files_in_dir2[j]->d_reclen == files_in_dir1[n]->d_reclen))
114                 flag = 0;
115         }
116
117         rewinddir (dir2);
118         if (flag) // Если 1
119         {
120             if (i == NPROC) // Если равен числу процессов
121             {
122                 pid_end = wait(NULL); // wait end process (завершение)
123                 for (m = 0; (m < NPROC) && (pids[m] != pid_end); m++); // Поиск ном. оконч. процесса
124                 cur = m;
125             } else cur = 1;
126
127             pids[cur] = fork(); // Порождается процесс
128             if (pids[cur] < 0) {
129                 perror("fork");
130                 return 1;
131             } else if (pids[cur] > 0) {
132                 in_str = str_dir1; // Присвоить назв. кат.
133                 in_str += "/"; // Добавить косую черту
134                 in_str += files_in_dir1[n]->d_name; // Присв. назв. файла
135                 out_str = str_dir2; // Присвоить назв. кат.
136                 out_str += "/"; // Добавить косую черту
137                 out_str += files_in_dir1[n]->d_name; // Присвоить назв. файла
138                 mycopyfile(in_str.c_str(), out_str.c_str()); // Копир.
139                 cout << "Pid = " << getpid() << " Name = " << files_in_dir1[n]->d_name << " Size = " << get_file_size(in_str) << endl;
140                 if (i < NPROC) i++;
141             }
142         }
143     }
144
145     closedir (dir1);
146     closedir (dir2);
147     for (i = 0; i < NPROC; i++)
148         waitpid (pids[i], NULL, 0);
149
150     return 0;
151 }
152

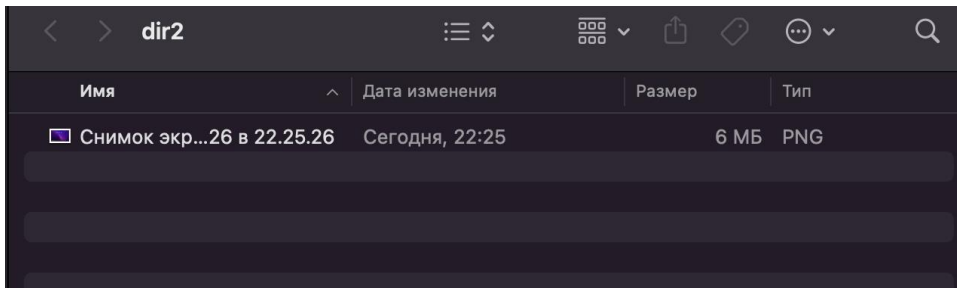
```

Dir1:



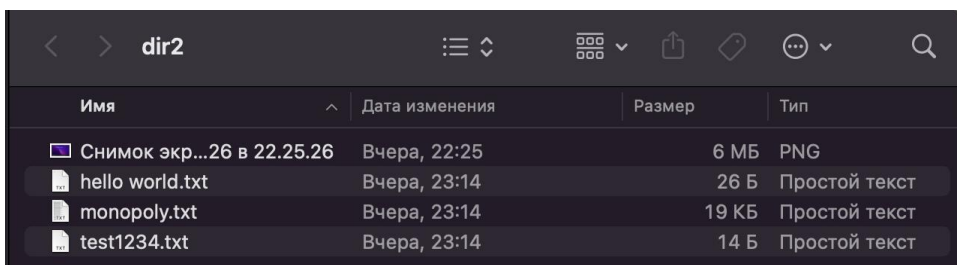
Имя	Дата изменения	Размер	Тип
hello world.txt	Сегодня, 18:25	26 Б	Простой текст
monopoly.txt	13 окт. 2022 г., 14:46	19 КБ	Простой текст
test1234.txt	13 окт. 2022 г., 15:02	14 Б	Простой текст

Dir2(до выполнения программы):



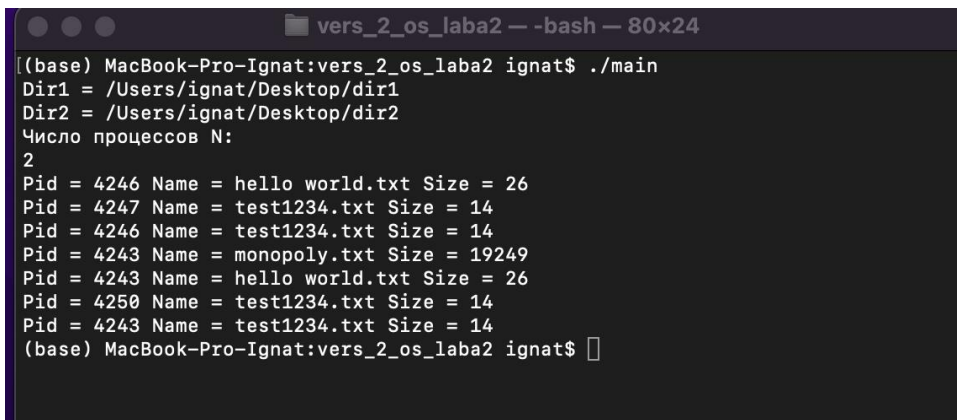
Имя	Дата изменения	Размер	Тип
Снимок экр...26 в 22.25.26	Сегодня, 22:25	6 МБ	PNG

Dir2(после выполнения программы):



Имя	Дата изменения	Размер	Тип
Снимок экр...26 в 22.25.26	Вчера, 22:25	6 МБ	PNG
hello world.txt	Вчера, 23:14	26 Б	Простой текст
monopoly.txt	Вчера, 23:14	19 КБ	Простой текст
test1234.txt	Вчера, 23:14	14 Б	Простой текст

Выход:



```

(base) MacBook-Pro-Ignat:vers_2_os_laba2 ignat$ ./main
Dir1 = /Users/ignat/Desktop/dir1
Dir2 = /Users/ignat/Desktop/dir2
Число процессов N:
2
Pid = 4246 Name = hello world.txt Size = 26
Pid = 4247 Name = test1234.txt Size = 14
Pid = 4246 Name = test1234.txt Size = 14
Pid = 4243 Name = monopoly.txt Size = 19249
Pid = 4243 Name = hello world.txt Size = 26
Pid = 4250 Name = test1234.txt Size = 14
Pid = 4243 Name = test1234.txt Size = 14
(base) MacBook-Pro-Ignat:vers_2_os_laba2 ignat$
  
```

Индивидуальное задание 2:

Написать программу, которая будет реализовывать следующие функции:

- сразу после запуска получает и сообщает свой ID и ID родительского процесса;
- перед каждым выводом сообщения об ID процесса и родительского процесса эта информация получается заново;
- порождает процессы, формируя генеалогическое дерево согласно варианту, сообщая, что "процесс с ID таким-то породил процесс с таким-то ID";
- перед завершением процесса сообщить, что "процесс с таким-то ID и таким-то ID родителя завершает работу";
- один из процессов должен вместо себя запустить программу, указанную в варианте задания.

На основании выходной информации программы предыдущего пункта изобразить генеалогическое дерево процессов (с указанием идентификаторов процессов). Объяснить каждое выведенное сообщение и их порядок в предыдущем пункте.

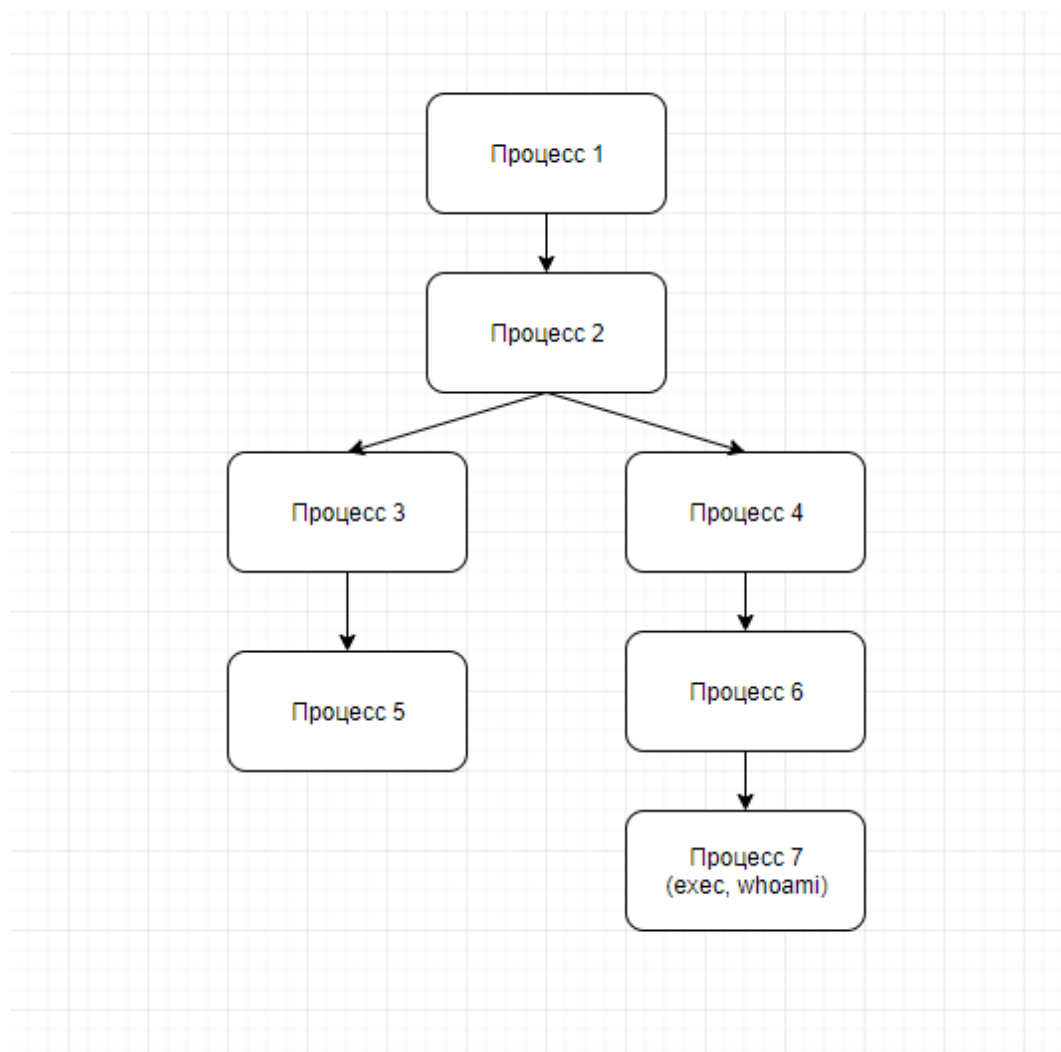
В столбце **fork** описано генеалогическое дерево процессов: каждая цифра указывает на относительный номер (не путать с `pid`) процесса, являющегося родителем для данного процесса. Например, строка `0 1 1 1 3` означает, что первый процесс не имеет родителя среди ваших процессов (порождается и запускается извне), второй, третий и четвертый - порождены первым, пятый - третьим.

В столбце `exes` указан номер процесса, выполняющего вызов **exes**, команды для которого указаны в последнем столбце. Запускайте команду обязательно с какими-либо параметрами.

Вариант: 14

№	fork	exes	
14	0 1 2 2 3 4 6	7	whoami

Дерево процессов:



Листинг:

```

1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/wait.h>
5 #include <stdlib.h>
6
7 int main(int argc, char* argv[])
8 {
9     pid_t pid;
10    printf("Процесс 1 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
11    if ((pid = fork()) == 0) {
12        printf("Процесс 2 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
13        if ((pid = fork()) == 0) {
14            printf("Процесс 3 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
15            if ((pid = fork()) == 0) {
16                printf("Процесс 5 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
17                printf("Процесс 5 с pid %d завершил работу\n", getpid());
18                exit(0);
19            }
20            wait(0);
21            printf("Процесс 3 с pid %d завершил работу\n", getpid());
22            exit(0);
23        }
24        wait(0);
25        if ((pid = fork()) == 0) {
26            printf("Процесс 4 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
27            if (fork() == 0) {
28                printf("Процесс 6 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
29                if ((pid = fork()) == 0) {
30                    printf("Процесс 7 создан, его pid: %d, его ppid: %d\n", getpid(), getppid());
31                    execl("/bin/whoami", "whoami", NULL);
32                    printf("Процесс 7 с pid %d завершил работу\n", getpid());
33                    exit(0);
34                }
35                wait(0);
36                printf("Процесс 6 с pid %d завершил работу\n", getpid());
37                exit(0);
38            }
39            wait(0);
40            printf("Процесс 4 с pid %d завершил работу\n", getpid());
41            exit(0);
42        }
43        wait(0);
44        printf("Процесс 2 с pid %d завершил работу\n", getpid());
45        exit(0);
46    }
47    wait(0);
48    printf("Процесс 1 с pid %d завершил работу\n", getpid());
49    exit(0);
50    return 0;
51

```

ВЫХОД:

```

/121702/Zayats/laba2/OS_2/ind2$ ./OS_2.exe
Процесс 1 создан, его pid: 10570, его ppid: 10546
Процесс 2 создан, его pid: 10571, его ppid: 10570
Процесс 3 создан, его pid: 10572, его ppid: 10571
Процесс 5 создан, его pid: 10573, его ppid: 10572
Процесс 5 с pid 10573 завершил работу
Процесс 3 с pid 10572 завершил работу
Процесс 4 создан, его pid: 10574, его ppid: 10571
Процесс 6 создан, его pid: 10575, его ppid: 10574
Процесс 7 создан, его pid: 10576, его ppid: 10575
perefurgon
Процесс 6 с pid 10575 завершил работу
Процесс 4 с pid 10574 завершил работу
Процесс 2 с pid 10571 завершил работу
Процесс 1 с pid 10570 завершил работу

```