Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4
по дисциплине: **ОСИСП**
Тема: GCC. ПРОЦЕССЫ

Выполнил
студент 2 курса
Корнасевич И. Д.

Проверил
Давидюк Ю. И.

Брест 2021

Задание для выполнения 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.

main.c

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

void created(int procNum){
    printf("Created %d: parent = %d, this = %d\n", procNum, getppid(), getpid());
}

void exited(int procNum){
    printf("Exited %d: parent = %d, this = %d\n", procNum, getppid(), getpid());
}


int main() {

    created(1);

    pid_t pid;
    int procNum;
    if ((pid = fork()) == 0){
        procNum = 2;
        created(procNum);

        if ((pid = fork()) == 0){
            procNum = 5;
            created(procNum);
            if ((pid = fork()) == 0){
                procNum = 6;
                created(procNum);
```

```
30                  exited(procNum);
31                  exit(0);
32              } else {
33                  sleep(1);
34              }
35              exited(procNum);
36              exit(0);
37          } else {
38              sleep(2);
39          }
40
41          exited(procNum);
42          exit(0);
43      } else {
44          sleep(3);
45      }
46
47      if ((pid = fork()) == 0){
48          procNum = 3;
49          created(procNum);
50          exited(procNum);
51          exit(0);
52      } else {
53          sleep(3);
54      }
55
56      if ((pid = fork()) == 0){
57          procNum = 4;
58          created(procNum);
59          if ((pid = fork()) == 0){
60              procNum = 7;
61              created(procNum);
62              execl("/bin/ls","ls","-a");
63              exited(procNum);
64              exit(0);
65          } else {
66              sleep(2);
67          }
68          exited(procNum);
69          exit(0);
70      } else {
71          sleep(3);
72      }
73      exited(1);
74  }
```

results.txt

```
1  Created 1: parent = 3605, this = 7089
2  Created 2: parent = 7089, this = 7090
3  Created 5: parent = 7090, this = 7091
4  Created 6: parent = 7091, this = 7092
5  Exited 6: parent = 7091, this = 7092
6  Exited 5: parent = 7090, this = 7091
7  Exited 2: parent = 7089, this = 7090
8  Created 3: parent = 7089, this = 7093
```

```
Exited 3: parent = 7089, this = 7093
Created 4: parent = 7089, this = 7095
Created 7: parent = 7095, this = 7096
.   CMakeCache.txt  cmake_install.cmake  lab1_OC.cbp  Testing
..  CMakeFiles      lab1_OC    Makefile
Exited 4: parent = 7089, this = 7095
Exited 1: parent = 3605, this = 7089
```

Дерево процессов:

- 1
  - 2
    - 5
      - 6
  - 3
  - 4
    - 7

Вывод: Я познакомился со способами управления жизненным циклом процессов при помощи функций fork(), execl(), exit(). Также узнал, что такое pid и ppid.