МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №4

«GCC. ПРОЦЕССЫ»

Выполнил:

Студент 2 курса
группы ПО-9
Мисиюк Алексей Сергеевич
(№ зач. книги 210664)

Проверила:

Давидюк Ю. И.

Брест 2022

Цель работы: изучить принцип работы с процессами в Unix подобных системах.

Ход работы

Вариант индивидуального задания № 6

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.

Индивидуальное задание

В столбце fork описано генеалогическое древо процессов: каждая цифра указывает на относительный номер (не путать с pid) процесса, являющегося родителем для данного процесса.

6. 0 1 1 2 4 4 4

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

6, ls

Код программы

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>

pid_t pid;
int relpid;

void child1();
void child2();
void child3();
void child4();
void child5();
void child6();

//0 1 1 2 4 4 4
//6, ls

int main()
{
        relpid = 0;
```

```
		printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());

		pid = fork();
		switch (pid) {
		 case -1:
			printf("Error doing fork() in procces %d", relpid);
			break;
		 case 0:
			child1();
			break;
		 default:
			pid = fork();
			switch (pid) {
			case -1:
				printf("Error doing fork() in procces %d", relpid);
				break;
			case 0:
				child2();
				break;
		 }
		}

		sleep(10-relpid);
		printf("Procces %d exit PID = %d; PPID = %d\n", relpid, getpid(), getppid());
		return 0;
}

void child1() {
		relpid = 1;
		printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());

		pid = fork();
		switch (pid) {
		case -1:
		printf("Error doing fork() in procces %d", relpid);
		break;
		case 0:
		child3();
		break;
		}
}

void child2() {
		relpid = 2;
```

```
		printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());
}

void child3() {
		relpid = 3;
		printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());

		pid = fork();
		switch (pid) {
		 case -1:
			printf("Error doing fork() in procces %d", relpid);
			break;
		 case 0:
			child4();
			break;
		 default:
			pid = fork();
			switch (pid) {
				case -1:
					printf("Error doing fork() in procces %d", relpid);
					break;
				case 0:
					child5();
					break;
				default:
					pid = fork();
					switch (pid) {
						case -1:
							printf("Error doing fork() in procces %d", relpid);
							break;
						case 0:
							child6();
							break;
					}
			}
		}
}

void child4() {
		relpid = 4;
		printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());
}

void child5() {
```

```
        relpid = 5;
        printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());

        printf("Procces %d exit PID = %d; PPID = %d\n", relpid, getpid(), getppid());
        execl("/bin/ls", "ls", "-l", NULL);
}

void child6() {
        relpid = 6;
        printf("Procces %d PID = %d; PPID = %d\n", relpid, getpid(), getppid());
}
```

Пример работы

```
$ ./bin/Release/report4
Procces 0 PID = 3722; PPID = 1989
Procces 1 PID = 3723; PPID = 3722
Procces 3 PID = 3724; PPID = 3723
Procces 5 PID = 3726; PPID = 3724
Procces 5 exit PID = 3726; PPID = 3724
Procces 6 PID = 3728; PPID = 3724
Procces 4 PID = 3725; PPID = 3724
Procces 2 PID = 3727; PPID = 3722
итого 24
drwxr-xr-x 3 aleksei aleksei 4096 окт 18 16:45 bin
-rw-r--r-- 1 aleksei aleksei 2795 окт 18 17:43 main.cpp
drwxr-xr-x 3 aleksei aleksei 4096 окт 18 16:45 obj
-rw-rw-r-- 1 aleksei aleksei  735 окт 18 16:40 report4.cbp
-rw-rw-r-- 1 aleksei aleksei  141 окт 18 17:19 report4.depend
-rw-rw-r-- 1 aleksei aleksei  361 окт 18 17:43 report4.layout
Procces 6 exit PID = 3728; PPID = 3724
Procces 4 exit PID = 3725; PPID = 3724
Procces 3 exit PID = 3724; PPID = 3723
Procces 2 exit PID = 3727; PPID = 3722
Procces 1 exit PID = 3723; PPID = 3722
Procces 0 exit PID = 3722; PPID = 1989
```

Вывод: изучен и опробован на языке С навык работы с родительскими и дочерними процессами в Unix подобных системах.