Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1 по курсу «Операционные системы»

Выполнил студент группы 921703 Кравцов М. С.

Проверил: Садовский М.Е.

МИНСК 2021

Тема: Создание процессов в ОС Linux

Цель: Изучить создание процессов в ОС Linux

Задание:
Написать программу, которая будет реализовывать следующие функции:
1) Сразу после запуска получает и сообщает свой ID и ID родительского процесса;
2) Перед каждым выводом сообщения об ID процесса и родительского процесса эта информация получается заново;
3) Порождает процессы, формируя генеалогическое дерево согласно варианту, сообщая, что "процесс с ID таким-то породил процесс с таким-то ID";
4) Перед завершением процесса сообщить, что "процесс с таким-то ID и таким-то ID родителя завершает работу";
5) Один из процессов должен вместо себя запустить программу, указанную в варианте задания.
6) 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.

Вариант 11:

№	fork	exec	
11	0 1 1 2 2 3 4	4	ls

Ход работы

Для порождения новых процессов использовалась функция fork() из стандартной библиотеки unistd.h. Данная функция создает новый процесс и возвращает одно из значений: -1 в случае ошибки, 0 для порожденного процесса, ID порожденного процесса для родительского процесса. С учетом этого возвращаемое значение проверялось на равенство 0. Бала использована функция wait() библиотеки sys/wait.h для ожидания завершения процесса потомка.

Для вывода информации об ID процесса (при завершении процесса, порождении нового процесса, запуске программы) использовались функций getpid() и getppid() из стандартной библиотеки unistd.h.

Для запуска другой программы в процессе использовалась одна из функций exec стандартной библиотеки unistd.h:

```
int execv(const char *pathname, char *const argv[]);
```

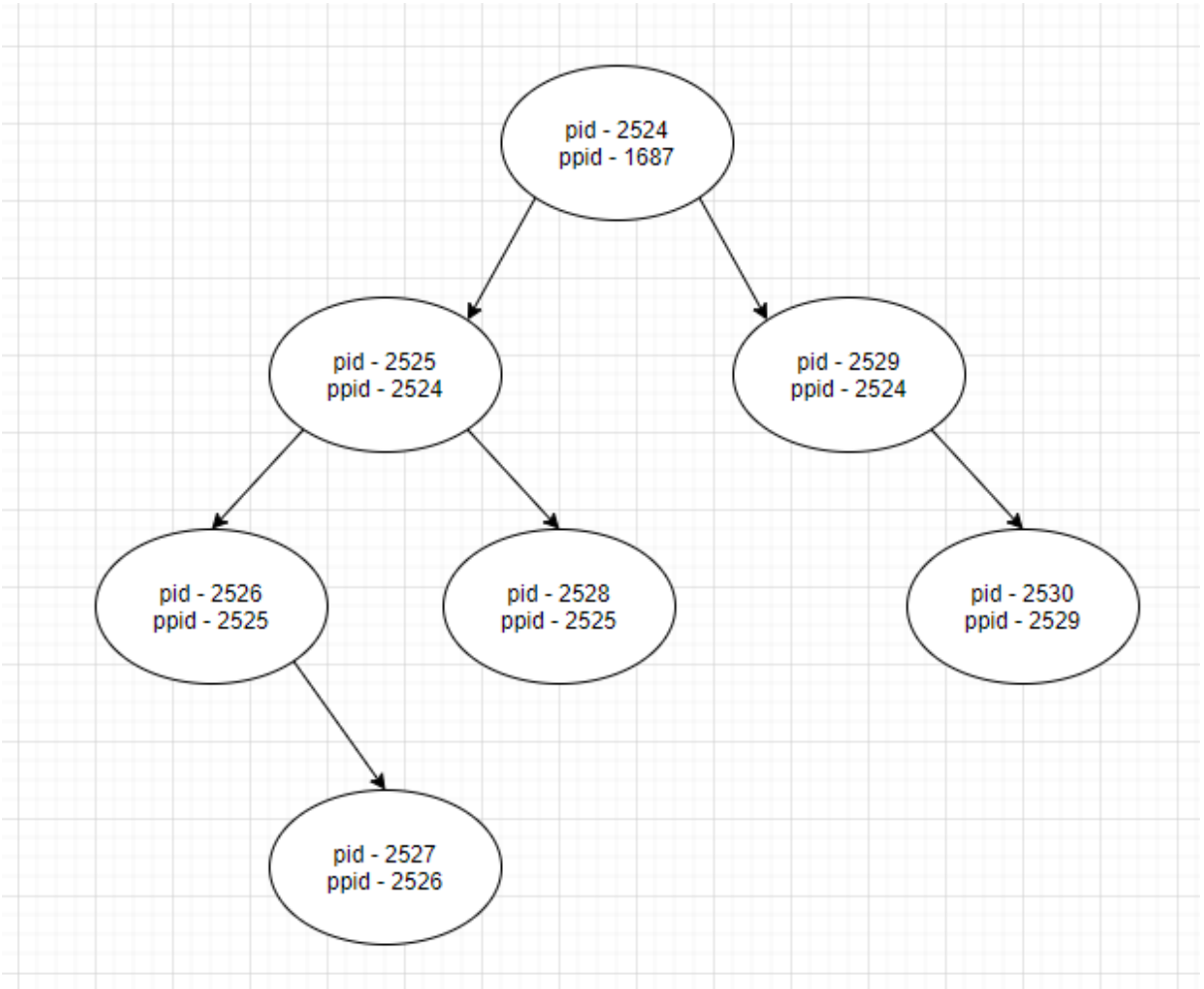
Выполняется программа ls с ключом –m.

Результат работы программы:

```
ID родительского процесса - 1687, ID процесса - 2524
процесс с ID-2524 породил процесс с ID-2525
процесс с ID-2525 породил процесс с ID-2526
процесс с ID-2526 породил процесс с ID-2527
процесс с ID-2527 и ID-2526 родителя завершает работу
итого 32
-rwxr-xr-x 1 mixa mixa 13040 вер 21 19:04 main
-rw-r--r-- 1 mixa mixa  2149 вер 21 19:04 main.c
-rw-r--r-- 1 mixa mixa  4248 вер 21 19:04 main.o
-rw-rw-r-- 1 mixa mixa    88 вер 18 14:41 Makefile
процесс с ID-2525 породил процесс с ID-2528
процесс с ID-2525 и ID-2524 родителя завершает работу
процесс с ID-2528 и ID-2525 родителя завершает работу
процесс с ID-2524 породил процесс с ID-2529
процесс с ID-2529 породил процесс с ID-2530
процесс с ID-2530 и ID-2529 родителя завершает работу
процесс с ID-2529 и ID-2524 родителя завершает работу
процесс с ID-2524 и ID-1687 родителя завершает работу
```

Изобразим генеалогическое дерево процессов:

0 1 1 2 2 3 4



Вывод

В данной лабораторной работе я изучил ряд полезных тем:
- метод создания процессов fork();
- завершение процессов, функция exit();
- семейство функций exec для изменения пользовательского контекста процессов;
- состояние процессов;

А также познакомился с терминологией системы Unix. Для этого был использован язык C и функционал его стандартных библиотек.