Домашнее задание 5

Реализовать базовый эмулятор командной строки. Программа работает в интерактивном режиме, ожидая команды от пользователя. При получении команды, создается новый процесс, который обрабатывает команду, а основной процесс дожидается ее выполнения, после чего предлагает пользователю ввести следующую команду.

Дедлайн: 2023-11-21

Терминал должен так уже уметь передавать аргументы командной строки в программу. Пример запуска

```
sapaeff@os2023:$ ./myterm
$ pwd
/home/sapaeff/os-2023/dilshod-sapaev
$ ls -a
.  ..  01  04
$ wc -l 01/sum.sh
21 01/sum.sh
$
```

Как создавать новые процессы?
- man 2 fork
- pid_t fork(); - это системный вызов который создает новый процесс с того момента, где он был вызван.
- Созданный процесс становится дочерним процессом для процесса, который вызвал его.
- В родительском и дочернем процессе возвращаются разные значения.

```
pid_t pid;
if ((pid = fork()) == 0 ) {
   printf("I am Child");
} else {
   printf("I am Parent"); // or error
}
```

Как вызывать команды?

- man 3 exec
- Семейство системных вызовов exec позволяют подменить текущий процесс на какой-то другой.
- В зависимости от выбранной функции можно передать аргументы процессу или задать переменный окружения.
- Функции НЕ возвращают управления при завершении без ошибки. Если функция завершилась успешно, программа завершается и вместо нее запускается другая программа
- Возврат из функции происходит только при возникновении ошибки.

Как дождаться выполнения дочернего процесса?

- man 3 waitpid
- Родительский процесс дожидается завершения (изменения состояния) конкретного дочернего процесса, после чего продолжает свою работу.
- pid (process id) - идентификатор процесса.
- Все процессы связаны в дерево вызовов.
  В корне дерева pid=0 (kernel)