Задание №1.
Создание и завершение процессов

19 марта 2016 г.

Каждый раз при открытии окна терминала в любой операционной системе запускается специальная программа — командный интерпретатор (оболочка). Например, для Windows это будет `cmd.exe`, для Linux и Mac OS X — `bash`. После запуска оболочка выводит приглашение командной строки (например символ `$`) и ожидает ввода команды с клавиатуры. Команда обычно включает в себя имя исполняемого файла, который следует запустить и аргументов командной строки, которые нужно этой программе передать.

Пример сеанса работы с оболочкой:

```
$ls
assignment1.aux assignment1.log assignment1.pdf assignment1.tex
$cal
    Марта 2016
вс пн вт ср чт пт сб
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
$nasm -f elf assignment.asm
$ld -o assignment assignment.o
```

После запуска на выполнение каждой очередной команды оболочка ждет пока команда отработает, выводит новое приглашение командной строки и ожидает ввода новой команды.

Задание: написать программу на языке C [1] с использование POSIX [2] системных вызовов, которая реализует базовый функционал оболочки. Программа должна в бесконечном цикле считывать команды пользователя и запускать соответствующие им программы. Завершать работу следует при возникновении ситуации «конец файла».

Указания: в ходе работы потребуется использовать системные вызовы `fork, execlp, wait`. Ознакомьтесь с их описанием с помощью справочного руководства `man`.

```
$man 2 fork
```

Для запуска исполняемого файла с помощью `execvp` необходимо подготовить структуру данных в виде массива строк, в которую записать имя программы и ее аргументы. Предлагается ввести ограничение на число аргументов в 16 и на максимальную длину каждого аргумента в 80 символов. Тогда эту структуру можно представить как двумерный символьный массив фиксированного размера.

```
char argv[16][80];
```

Запуск внешней программы в POSIX реализуется с помощью связки `fork+exec`. Ниже приведен пример запуска программы `ls` с аргументами `-l` и `-a`.

```
/* Exec ls -l -a */

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

int main()
{
  pid_t pid = fork();
  if (!pid) { // child branch
    int rv = execlp("ls", "ls", "-l", "-a", NULL);
    if (rv == -1) {
      perror("execlp");
      return EXIT_FAILURE;
    }
  }
  // parent branch
  pid = wait(NULL);
  if (pid == -1) {
    perror("wait");
    return EXIT_FAILURE;
  }
  return EXIT_SUCCESS;
}
```

Результаты данной работы станут основой для выполнения дальнейших заданий. В связи с этим предлагается использовать для разработки систему контроля версий (например `git` [3]) и хранить результаты работ в удаленном репозитории (например `github.com`).

- Создайте учетную запись на `http://github.com`.
- Создайте новый репозиторий (например `OSLabs`).

- Добавьте в репозиторий код решения.
- Пришлите ссылку на репозиторий для проверки.

Список литературы

[1] Б. Керниган, Д. Ритчи. Язык программирования C, 2-е изд. — Москва: Вильямс, 2015. — 288 с. — ISBN 978-5-8459-1976-5.

[2] А. Столяров. Введение в операционные системы, М.: Изд. отдел ВМК МГУ, 2006. - ISBN 5-89407-246-8. `http://stolyarov.info/books/osintro`

[3] `https://git-scm.com/book/ru/v1`