Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работ №2 по курсу
«Операционные системы»

Управление процессами в ОС

Студент: Болдырев Антон Константинович
Группа: М8О-206Б-18
Вариант: 6
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Содержание

Постановка задачи.

На вход программе подаётся команда интерпретатора команд. Программа должна произвести вывод команды заменяя знаки табуляции и space на знаки «____» и «_» соответственно.

Общие сведения о программе

Программа компилируется из файла main.c. В программе используются следующие системные вызовы:

1. **read** – для чтения данных из pipe.
2. **write** – для записи данных в pipe.
3. **pipe** – для создания однонаправленного канала, через который из дочернего процесса данные передаются родительскому. Возвращает два дескриптора файлов: для чтения и для записи.
4. **fork** – для создания дочернего процесса.
5. **close** – для закрытия pipe после окончания считывания результата выполнения команды.

Общий метод и алгоритм решения.

С помощью fork() создаётся дочерний процесс. Для обмена данными между дочерним и родительским процессами создаём pipe. Если pid равен нулю, то идёт выполнение дочернего процесса. В неё мы используем dup2(), где перенаправляем стандартный поток вывода в pipe. Далее с помощью execvp() выполняется команда интерпретатора команд, в результате её работы попадает в pipe. Родительский процесс использует waitpid(), чтобы начать своё выполнение после дочернего. Далее из pipe() по одному перебираются все символы. Если это пробел, то он переводится в «_». После этого результат работы выводится в стандартный поток вывода.

Файлы программы.

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main(int argc, char** argv)
{
   pid_t pid;
   int rv;
   int fd[2];
   if (pipe(fd) == -1) {
      perror("pipe error");
      exit(1);
   }
   switch ( pid = fork() ) {
      case -1:
         perror("fork");
         exit(1);
```

```
        case 0:
            close(fd[0]);                   // потомок не читает
            dup2(fd[1], STDOUT_FILENO);       // перенаправление stdout
            rv = execvp(argv[1], argv + 1);
            exit(0);

        default:
            waitpid(pid, &rv, 0);
            close(fd[1]);                   // родитель не записывает
            char buf[1];
            while(read(fd[0], buf, 1) > 0) {
                if (buf[0] == ' ') {
                    buf[0] = '_';   //замена замена пробелов на нижнее подчеркивание
                }
                write(1, buf, 1);            // в stdout
            }
            exit(WEXITSTATUS(rv));
            wait(0);
    }
return 0;
}
```

Демонстрация работы программы.

```
anton@anton-Lenovo-ideapad-320-15IKB:~/OS/OS_Lab_2_1.1$ ./a.out cat main.c | head -
50
#include_<stdio.h>
#include_<stdlib.h>
#include_<errno.h>
#include_<unistd.h>
#include_<sys/types.h>
#include_<sys/wait.h>
int_main(int_argc,_char**_argv)
{
____pid_t_pid;
____int_rv;
____int_fd[2];
____if_(pipe(fd)_==_-1)_{
________perror("pipe_error");
________exit(1);
____}
____switch_(_pid_=_fork()_)_{
________case_-1:
____________perror("fork");
____________exit(1);

________case_0:
```

```
____________close(fd[0]);_____________________//_потомок_не_читает
____________dup2(fd[1],_STDOUT_FILENO);_______//_перенаправление_stdout
____________rv_=_execvp(argv[1],_argv_+_1);
____________exit(0);

________default:
____________waitpid(pid,_&rv,_0);
____________close(fd[1]);_____________________//_родитель_не_записывает
____________char_buf[1];
____________while(read(fd[0],_buf,_1)_>_0)_{
________________if_(buf[0]_==_'_')_{
____________________buf[0]_=_'_';___//
замена_замена_пробелов_на_нижнее_подчеркивание
________________}
________________write(1,_buf,_1);_____________//_в_stdout
____________}
____________exit(WEXITSTATUS(rv));
____________wait(0);
____}
return_0;
}
anton@anton-Lenovo-ideapad-320-15IKB:~/OS/OS_Lab_2_1.1$ ./a.out man ls
LS(1)_____________________________User_Commands_____________________________LS(1)

NAME
_______ls_-_list_directory_contents

SYNOPSIS
_______ls_[OPTION]..._[FILE]...

DESCRIPTION
_______List__information__about__the_FILEs_(the_current_directory_by_default).
_______Sort_entries_alphabetically_if_none_of_-cftuvSUX_nor_--sort__is__speci-
_______fied.

_______Mandatory__arguments__to__long__options_are_mandatory_for_short_options
_______too.

_______-a,_--all
_____________do_not_ignore_entries_starting_with_.

_______-A,_--almost-all
_____________do_not_list_implied_._and_..

_______--author
_____________with_-l,_print_the_author_of_each_file

_______-b,_--escape
_____________print_C-style_escapes_for_nongraphic_characters

_______--block-size=SIZE
_____________scale_sizes_by_SIZE_before_printing_them;_e.g.,_'--block-size=M'
_____________prints_sizes_in_units_of_1,048,576_bytes;_see_SIZE_format_below

            ...
```

Вывод

Я научился создавать процессы, используя системный вызов fork(). Также я обрел навыки взаимодействия между процессами с помощью pipe(), получил новые знания о файловых дескрипторах и выполнении команд интерпретатора команд с помощью семейства системных вызовов exec().