Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М80-206Б-20

Студент: Голубев Т.Д.

Преподаватель: Миронов Е.С.

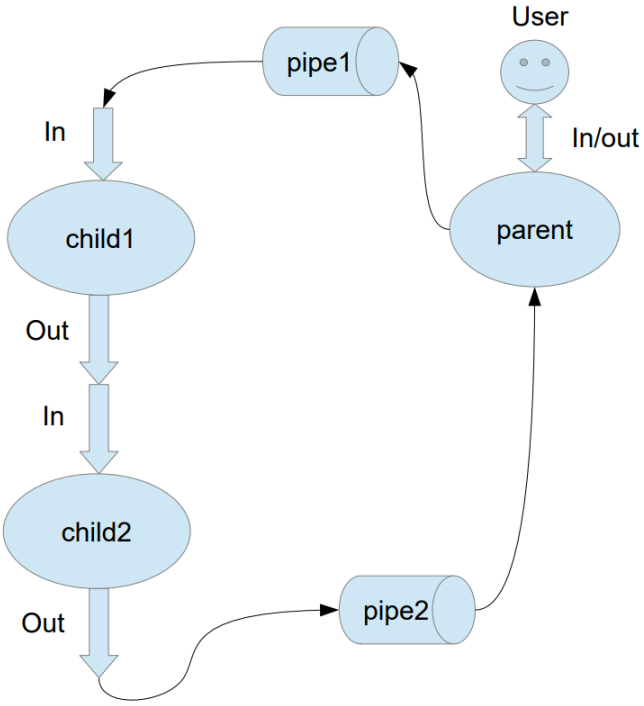
Оценка: _____

Дата: 21.09.2023

Москва, 2023

Постановка задачи

Вариант 11.



Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода.

Child1 переводит строки в верхний регистр. Child2 превращает все пробельные символы в символ «_».

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создаёт дочерний процесс.
- int pipe(int *fd); – создаёт канал (пайп).
- int dup2(int oldfd, int newfd); – делает newfd копией дескриптора oldfd, закрывая newfd, если требуется.
- int execl(const char *path, const char *arg, ...); – заменяет текущий образ процесса новым образом процесса.
- int close(int fd); – закрывает файловый дескриптор.
- ssize_t write(int fd, const void *buf, size_t count); – записывает до count байтов из буфера buf в файл, на который ссылается файловый описатель fd.
- ssize_t read(int fd, void *buf, size_t count); – пытается записать count байтов файлового описателя fd в буфер, адрес которого начинается с buf.

Создал три пайпа для связи дочерних процессов и родительского с дочерними с помощью pipe(). Далее создал два дочерних процесса с помощью fork() и вызвал скомпилированные child1.cpp и child2.cpp с помощью execl(). В родительском процессе читал символы, которые пишет пользователь и посылал в child1. Первый дочерний процесс с помощью toupper() «озаглавливал» символы и посылал их child2. Второй дочерний процесс заменял пробел на «_» и посылал обратно родительскому процессу, который в свою очередь выводил их на стандартный вывод.

Код программы

main.cpp

```
#include <unistd.h>
#include <iostream>
#include <string>
#include <cctype>

int create_process() {
    pid_t pid = fork();
    if (pid == -1) {
        perror("Fork error!\n");
        exit(-1);
    }
    return pid;
}

void create_pipe(int* pipe_fd) {
    if (pipe(pipe_fd) == -1) {
        perror("Pipe error!\n");
        exit(-1);
    }
}

void dup_fd(int oldfd, int newfd) {
    if (dup2(oldfd, newfd) == -1) {
        perror("dup2 error!\n");
        exit(-1);
    }
}

int main() {
    int pipe1_fd[2], pipe2_fd[2];   // pipe1 - from parent to child1, pipe2 - from
child2 to parent
    create_pipe(pipe1_fd);
    create_pipe(pipe2_fd);

    pid_t child1 = create_process();
    if (child1 == 0) {
        close(pipe1_fd[1]);
        close(pipe2_fd[0]);

        int pipech_fd[2];
        create_pipe(pipech_fd);

        pid_t child2 = create_process();

        if (child2 == 0) { // child2
            close(pipech_fd[0]);
            close(pipe2_fd[1]);

            dup_fd(pipe1_fd[0], STDIN_FILENO);
            dup_fd(pipech_fd[1], STDOUT_FILENO);
```

```
            execl("../build/child2", "../build/child2", NULL);

            close(pipech_fd[1]);
            close(pipe1_fd[0]);
        } else {    // child1
            close(pipe1_fd[0]);
            close(pipech_fd[1]);

            dup_fd(pipech_fd[0], STDIN_FILENO);
            dup_fd(pipe2_fd[1], STDOUT_FILENO);

            execl("../build/child1", "../build/child1", NULL);

            close(pipe1_fd[0]);
            close(pipe2_fd[1]);
        }
    } else {    // parent
        close(pipe1_fd[0]);
        close(pipe2_fd[1]);
        char c = getchar();
        while (c != EOF) {
            write(pipe1_fd[1], &c, sizeof(c));
            read(pipe2_fd[0], &c, sizeof(c));

            putchar(c);

            c = getchar();
        }

        close(pipe1_fd[1]);
        close(pipe2_fd[0]);
    }

    return 0;
}
```

child1.cpp

```
#include <iostream>
#include <unistd.h>

int main() {
    char c;
    while (read(STDIN_FILENO, &c, sizeof(c)) != -1) {
        c = toupper(c);
        write(STDOUT_FILENO, &c, sizeof(c));
    }
    close(STDIN_FILENO);
    close(STDOUT_FILENO);

    return 0;
}
```

child2.cpp

```
#include <iostream>
#include <unistd.h>

int main() {
    char c;
    while (read(STDIN_FILENO, &c, sizeof(c)) != -1) {
        if (c == ' ') {
            c = '_';
```

```
            }
            write(STDOUT_FILENO, &c, sizeof(c));
        }
        close(STDIN_FILENO);
        close(STDOUT_FILENO);

        return 0;
    }
```

Протокол работы программы

Тестирование:

```
cat_mood@nuclear-box:~/programming/mai-os-labs/lab01/build$ ./main
> hello world!
HELLO_WORLD!
> HaaH hAAh
HAAH_HAAH______
>

____
> 123 #$$ {}":
123_#$$_{}":
```

Strace:

```
execve("./main", ["./main"], 0x7ffed87e1f80 /* 36 vars */) = 0
brk(NULL)                               = 0x564a0cccc000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff632e3370) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe34df35000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=18023, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 18023, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe34df30000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe34dd04000
mprotect(0x7fe34dd9e000, 1576960, PROT_NONE) = 0
mmap(0x7fe34dd9e000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x9a000) = 0x7fe34dd9e000
mmap(0x7fe34deaf000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ab000) =
0x7fe34deaf000
mmap(0x7fe34df1f000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x21a000) = 0x7fe34df1f000
```

```
      mmap(0x7fe34df2d000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fe34df2d000

      close(3)                                  = 0

      openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

      read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

      pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

      pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

      pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0i8\235HZ\227\223\333\350s\360\352,\223\340."..., 68,
896) = 68

      newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2216304, ...}, AT_EMPTY_PATH) = 0

      pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

      mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe34dadc000

      mmap(0x7fe34db04000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7fe34db04000

      mmap(0x7fe34dc99000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7fe34dc99000

      mmap(0x7fe34dcf1000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x214000) = 0x7fe34dcf1000

      mmap(0x7fe34dcf7000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fe34dcf7000

      close(3)                                  = 0

      openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

      read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

      newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0

      mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe34d9f5000

      mmap(0x7fe34da03000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xe000) = 0x7fe34da03000

      mmap(0x7fe34da7f000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) =
0x7fe34da7f000

      mmap(0x7fe34dada000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xe4000) = 0x7fe34dada000

      close(3)                                  = 0

      openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3

      read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

      newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0

      mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fe34d9d5000

      mmap(0x7fe34d9d8000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000)
= 0x7fe34d9d8000

      mmap(0x7fe34d9ef000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) =
0x7fe34d9ef000

      mmap(0x7fe34d9f3000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1d000) = 0x7fe34d9f3000

      close(3)                                  = 0
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe34d9d3000
arch_prctl(ARCH_SET_FS, 0x7fe34d9d43c0) = 0
set_tid_address(0x7fe34d9d4690)         = 30207
set_robust_list(0x7fe34d9d46a0, 24)     = 0
rseq(0x7fe34d9d4d60, 0x20, 0, 0x53053053) = 0
mprotect(0x7fe34dcf1000, 16384, PROT_READ) = 0
mprotect(0x7fe34d9f3000, 4096, PROT_READ) = 0
mprotect(0x7fe34dada000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe34d9d1000
mprotect(0x7fe34df1f000, 45056, PROT_READ) = 0
mprotect(0x564a0bd3b000, 4096, PROT_READ) = 0
mprotect(0x7fe34df6f000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fe34df30000, 18023)           = 0
getrandom("\x89\x5c\xc9\x60\xf1\x95\xed\x7f", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x564a0cccc000
brk(0x564a0cced000)                     = 0x564a0cced000
futex(0x7fe34df2d77c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fe34d9d4690) = 30208
close(3)                                = 0
close(6)                                = 0
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}, AT_EMPTY_PATH) = 0
read(0, hello world
"hello world\n", 1024)          = 12
write(4, "h", 1)                        = 1
read(5, "H", 1)                         = 1
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}, AT_EMPTY_PATH) = 0
write(4, "e", 1)                        = 1
read(5, "E", 1)                         = 1
write(4, "l", 1)                        = 1
read(5, "L", 1)                         = 1
write(4, "l", 1)                        = 1
read(5, "L", 1)                         = 1
write(4, "o", 1)                        = 1
read(5, "O", 1)                         = 1
```

```
write(4, " ", 1)                        = 1
read(5, "_", 1)                         = 1
write(4, "w", 1)                        = 1
read(5, "W", 1)                         = 1
write(4, "o", 1)                        = 1
read(5, "O", 1)                         = 1
write(4, "r", 1)                        = 1
read(5, "R", 1)                         = 1
write(4, "l", 1)                        = 1
read(5, "L", 1)                         = 1
write(4, "d", 1)                        = 1
read(5, "D", 1)                         = 1
write(4, "\n", 1)                       = 1
read(5, "\n", 1)                        = 1
write(1, "HELLO_WORLD\n", 12HELLO_WORLD
)           = 12
read(0, "", 1024)                       = 0
close(4)                                = 0
close(5)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе лабораторной работы я написал программу, которая делает системные вызовы. Я научился использовать пайпы и работать с процессами. В ходе работы я столкнулся с некоторыми проблемами: бесконечный цикл (решил заменой cin/cout на read/write) и segmentation fault (решил использованием char вместо std::string).