

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: М8О-209БВ-24

Студент: Крысанов А.Ю.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 30.11.25

Москва, 2025

Постановка задачи

Вариант 1.

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 пересыпает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересыпает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода. Child1 переводит строки в верхний регистр. Child2 убирает все задвоенные пробелы

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void) - создает дочерний процесс, который является копией родительского процесса.
 - int pipe(int *fd) - создает канал и возвращает два дескриптора
 - int close(int fd) - закрывает файловый дескриптор
 - int dup2(int oldfd, int newfd) – перенаправляет стандартные потоки ввода-вывода
 - int execl(const char *path, const char *arg, ...) - загружает и запускает новую программу, полностью заменяя текущий процесс
 - pid_t waitpid(pid_t pid, int* status, int options) – ожидает завершения дочернего процесса.
-
- Родительский процесс создает три канала: pipe1 для передачи данных child_1, pipe2 для передачи данных от child_1 к child_2, pipe3 для передачи результата обратно родителю.
 - Родитель создает процесс child_1. В этом процессе стандартный ввод перенаправляется на pipe1, стандартный вывод — на pipe2, после чего запускается программа child_1, которая переводит полученную строку в верхний регистр и отправляет её далее.
 - Затем родитель создает процесс child_2. В этом процессе стандартный ввод перенаправляется на pipe2, стандартный вывод — на pipe3, после чего запускается программа child_2, которая принимает строку от child_1 и удаляет задвоенные пробелы.
 - Родительский процесс читает строку пользователя, записывает её в pipe1, затем читает окончательный результат из pipe3.
 - Полученная обработанная строка выводится родителем в стандартный вывод.
 - После завершения обмена данными родитель закрывает каналы, ожидает завершение обоих дочерних процессов и завершает программу.

Код программы

parent.c

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

```
#define BUFFER_SIZE 256
```

```
int main()
{
    char buffer[BUFFER_SIZE];
    int pipe1[2], pipe2[2], pipe3[2];

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1 || pipe(pipe3) == -1) {
        perror("pipe failed");
        exit(EXIT_FAILURE);
    }
```

```
/*
pipe1: parent -> child_1
pipe2: child_1 -> child_2
pipe3: child_2 -> parent
*/
```

```
// дочерний процесс child_1
pid_t child_1 = fork();

if (child_1 == 0) {
```

```
close(pipe1[1]); close(pipe2[0]); close(pipe3[0]); close(pipe3[1]);  
dup2(pipe1[0], STDIN_FILENO);  
dup2(pipe2[1], STDOUT_FILENO);  
execl("./child_1", "child_1", NULL);  
perror("execl child_1 failed");  
exit(EXIT_FAILURE);  
}
```

```
// дочерний процесс child_2  
pid_t child_2 = fork();  
if (child_2 == 0) {  
    close(pipe2[1]); close(pipe3[0]); close(pipe1[0]); close(pipe1[1]);  
    dup2(pipe2[0], STDIN_FILENO);  
    dup2(pipe3[1], STDOUT_FILENO);  
    execl("./child_2", "child_2", NULL);  
    perror("execl child_2 failed");  
    exit(EXIT_FAILURE);  
}
```

```
close(pipe1[0]); close(pipe2[0]); close(pipe2[1]); close(pipe3[1]);
```

```
//родительский процесс parent  
printf("Введите текст для обработки: ");  
if (fgets(buffer, BUFFER_SIZE, stdin) == NULL) {  
    printf("Ошибка ввода!\n");  
    exit(EXIT_FAILURE);  
}  
buffer[strcspn(buffer, "\n")] = '\0';
```

```
write(pipe1[1], buffer, strlen(buffer) + 1);
```

```

close(pipe1[1]);

char result[BUFFER_SIZE];
ssize_t bytes_read = read(pipe3[0], result, BUFFER_SIZE - 1);

if (bytes_read == -1) {
    perror("Ошибка чтения из pipe3");
    exit(EXIT_FAILURE);
} else if (bytes_read == 0) {
    printf("pipe3 закрыт, данных нет\n");
} else {
    result[bytes_read] = '\0';
    printf("Получен результат из child_2: %s\n", result);
}

close(pipe3[0]);

waitpid(child_1, NULL, 0);
waitpid(child_2, NULL, 0);

printf("Программа завершена.\n");

return 0;
}

```

child1.c

```

#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <unistd.h>

```

```

#define BUFFER_SIZE 256

int main()
{
    char buffer[BUFFER_SIZE];
    ssize_t bytes_read = read(STDIN_FILENO, buffer, BUFFER_SIZE - 1);

    if (bytes_read > 0) {
        buffer[bytes_read] = '\0';

        for (int i = 0; buffer[i] != '\0'; i++) {
            buffer[i] = toupper(buffer[i]);
        }

        write(STDOUT_FILENO, buffer, strlen(buffer) + 1);
    }

    return 0;
}

```

child2.c

```

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <unistd.h>

#define BUFFER_SIZE 256

```

```

int main() {

```

```
char buffer[BUFFER_SIZE];
char result[BUFFER_SIZE];
ssize_t bytes_read;

bytes_read = read(STDIN_FILENO, buffer, BUFFER_SIZE - 1);

if (bytes_read > 0) {
    buffer[bytes_read] = '\0';

    int j = 0;
    int prev_space = 0;

    for (int i = 0; buffer[i] != '\0'; i++) {
        if (isspace(buffer[i])) {
            if (!prev_space) {
                result[j++] = ' ';
                prev_space = 1;
            }
        } else {
            result[j++] = buffer[i];
            prev_space = 0;
        }
    }
    result[j] = '\0';

    write(STDOUT_FILENO, result, strlen(result) + 1);
}

return 0;
}
```

Протокол работы программы

```
root → /workspace/laba1 $ ./parent
```

Введите текст для обработки: hi my name is pedri and i am a professional football player

Получен результат из child_2: 'HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL PLAYER'

Программа завершена.

Strace:

```
root → /workspace/laba1 $ strace -f -s 1000 ./parent
execve("./parent", ["/./parent"], 0x7ffc20d169e8 /* 12 vars */) = 0
brk(NULL)                      = 0xae93000
brk(0xae93c40)                 = 0xae93c40
arch_prctl(ARCH_SET_FS, 0xae93300) = 0
uname({sysname="Linux", nodename="0e4263317ea3", ...}) = 0
readlink("/proc/self/exe", "/workspace/laba1/parent", 4096) = 23
brk(0xaeb4c40)                 = 0xaeb4c40
brk(0xaeb5000)                 = 0xaeb5000
mprotect(0x4b1000, 12288, PROT_READ) = 0
pipe([3, 4])                   = 0
pipe([5, 6])                   = 0
pipe([7, 8])                   = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD) = 50
Process 50 attached, child_tidptr=0xae935d0
[pid 49] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid 50] close(4)                  = 0
[pid 50] close(5) strace: Process 51 attached<unfinished ...>
[pid 49] <... clone resumed>, child_tidptr=0xae935d0) = 51
[pid 49] close(3 <unfinished ...>
[pid 50] <... close resumed>)      = 0
[pid 49] <... close resumed>)      = 0
[pid 51] close(6 <unfinished ...>
[pid 49] close(5 <unfinished ...>
[pid 50] close(7 <unfinished ...>
[pid 49] <... close resumed>)      = 0
```

```
[pid  51] <... close resumed>          = 0
[pid  49] close(6 <unfinished ...>
[pid  50] <... close resumed>          = 0
[pid  49] <... close resumed>          = 0
[pid  51] close(7 <unfinished ...>
[pid  49] close(8 <unfinished ...>
[pid  50] close(8 <unfinished ...>
[pid  49] <... close resumed>          = 0
[pid  51] <... close resumed>          = 0
[pid  50] <... close resumed>          = 0
[pid  51] close(3)                      = 0
[pid  50] dup2(3, 0 <unfinished ...>
[pid  51] close(4 <unfinished ...>
[pid  50] <... dup2 resumed>          = 0
[pid  51] <... close resumed>          = 0
[pid  51] dup2(5, 0 <unfinished ...>
[pid  50] dup2(6, 1 <unfinished ...>
[pid  51] <... dup2 resumed>          = 0
[pid  50] <... dup2 resumed>          = 1
[pid  51] dup2(8, 1)                  = 1
[pid  50] execve("./child_1", ["child_1"], 0x7fffcf1f8738 /* 12 vars */ <unfinished ...>
[pid  51] execve("./child_2", ["child_2"], 0x7fffcf1f8738 /* 12 vars */ <unfinished ...>
[pid  49] fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
[pid  49] fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
[pid  49] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265\321\202\320\265\320\272\321\201\321\202 \320\264\320\273\321\217\320\276\320\261\321\200\320\260\320\261\320\276\321\202\320\272\320\270: ", 53Введите текст для обработки: ) = 53
[pid  49] read(0, <unfinished ...>
[pid  50] <... execve resumed>          = 0
[pid  51] <... execve resumed>          = 0
[pid  50] brk(NULL)                    = 0x30700000
[pid  50] brk(0x30700c40)            = 0x30700c40
```

```
[pid  51] brk(NULL)                      = 0x20f4f000
[pid  51] brk(0x20f4fc40)                  = 0x20f4fc40
[pid  50] arch_prctl(ARCH_SET_FS, 0x30700300) = 0
[pid  50] uname({sysname="Linux", nodename="0e4263317ea3", ...}) = 0
[pid  51] arch_prctl(ARCH_SET_FS, 0x20f4f300 <unfinished ...>
[pid  50] readlink("/proc/self/exe", <unfinished ...>
[pid  51] <... arch_prctl resumed>)      = 0
[pid  51] uname(<unfinished ...>
[pid  50] <... readlink resumed>"/workspace/laba1/child_1", 4096) = 24
[pid  51] <... uname resumed>{sysname="Linux", nodename="0e4263317ea3", ...}) = 0
[pid  50] brk(0x30721c40)                  = 0x30721c40
[pid  50] brk(0x30722000)                  = 0x30722000
[pid  51] readlink("/proc/self/exe", <unfinished ...>
[pid  50] mprotect(0x4af000, 12288, PROT_READ <unfinished ...>
[pid  51] <... readlink resumed>"/workspace/laba1/child_2", 4096) = 24
[pid  50] <... mprotect resumed>)        = 0
[pid  51] brk(0x20f70c40)                  = 0x20f70c40
[pid  51] brk(0x20f71000)                  = 0x20f71000
[pid  50] read(0, <unfinished ...>
[pid  51] mprotect(0x4af000, 12288, PROT_READ) = 0
[pid  51] read(0, hi my name is pedri and i am a professional football player
              <unfinished ...>
[pid  49] <... read resumed>"hi my name is pedri and i am a professional football
player\n", 1024) = 63
[pid  49] write(4, "hi my name is pedri and i am a professional football
player\0", 63) = 63
[pid  50] <... read resumed>"hi my name is pedri and i am a professional football
player\0", 255) = 63
[pid  49] close(4 <unfinished ...>
[pid  50] write(1, "HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL
PLAYER\0", 63 <unfinished ...>
[pid  49] <... close resumed>)          = 0
[pid  49] read(7, <unfinished ...>
[pid  50] <... write resumed>)          = 63
```

```
[pid 51] <... read resumed>"HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL  
PLAYER\0", 255) = 63

[pid 51] write(1, "HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL PLAYER\0",  
60 <unfinished ...>

[pid 49] <... read resumed>"HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL  
PLAYER\0", 255) = 60

[pid 51] <... write resumed> = 60

[pid 49] write(1, "\320\237\320\276\320\273\321\203\321\207\320\265\320\275  
\321\200\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 \320\270\320\267  
child_2: 'HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL PLAYER'\n", 110Получен  
результат из child_2: 'HI MY NAME IS PEDRI AND I AM A PROFESSIONAL FOOTBALL PLAYER'  
 ) = 110

[pid 49] close(7) = 0

[pid 49] wait4(50, <unfinished ...>

[pid 50] exit_group(0 <unfinished ...>

[pid 51] exit_group(0 <unfinished ...>

[pid 50] <... exit_group resumed> = ?

[pid 51] <... exit_group resumed> = ?

[pid 50] +++ exited with 0 +++

[pid 51] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 50

--- SIGCHLD {si_signo=SIGHLD, si_code=CLD_EXITED,  
si_pid=50, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

wait4(51, NULL, 0, NULL) = 51

write(1,  
"\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\3  
20\260  
\320\267\320\260\320\262\320\265\321\200\321\210\320\265\320\275\32  
0\260.\n", 39Программа завершена.

) = 39

exit_group(0) = ?

+++ exited with 0 +++
```

Выход

В ходе выполнения лабораторной работы я научился создавать пайпы, процессы, понял, как программа взаимодействует с процессами, научился перенаправлять потоки ввода/вывода, то есть изучил инструменты, которые помогают работать напрямую с компьютером, что, по-моему, очень важно. Основная сложность была в правильном закрытии файловых дескрипторов – если они оставались открытыми, программа зависала. Лабораторная работа понравилась, теперь стало чуть понятнее, как процессы работают вместе, взаимодействуют, как передают друг другу информацию.