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-211Б-23

Студент: Грищенко В.А.

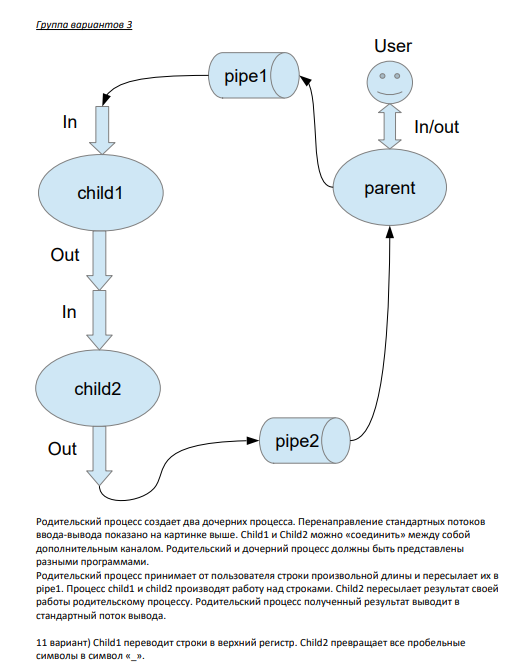
Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 29.10.24

Москва, 2024

**Постановка задачи**

**Вариант 11.**

**Общий метод и алгоритм решения**

Использованные системные вызовы:

1. **pid\_t fork(void);** – создает дочерний процесс.
2. **int pipe(int \*fd);** – создает канал (pipe) для межпроцессного взаимодействия.
3. **int dup2(int oldfd, int newfd);** – дуплицирует файловый дескриптор.
4. **int execl(const char \*path, const char \*arg, ...);** – заменяет текущий процесс новым процессом.
5. **int close(int fd);** – закрывает файловый дескриптор.
6. **ssize\_t read(int fd, void \*buf, size\_t count);** – читает данные из файлового дескриптора.
7. **ssize\_t write(int fd, const void \*buf, size\_t count);** – записывает данные в файловый дескриптор.
8. **pid\_t waitpid(pid\_t pid, int \*status, int options);** – ожидает завершения дочернего процесса.

**Описание лабораторной работы**

В рамках лабораторной работы была разработана программа, которая демонстрирует межпроцессное взаимодействие с использованием каналов (pipes) и системных вызовов. Программа состоит из родительского процесса и двух дочерних процессов, которые обрабатывают данные последовательно.

**Цель лабораторной работы**

Целью лабораторной работы было изучение и применение системных вызовов для создания и управления процессами, а также использование каналов для межпроцессного взаимодействия.

**Описание программы**

Программа состоит из трех частей:

1. **Родительский процесс (parent.c)**: Считывает данные из стандартного ввода (stdin), передает их первому дочернему процессу через канал, затем передает обработанные данные второму дочернему процессу и выводит окончательный результат в стандартный вывод (stdout).
2. **Первый дочерний процесс (child1.c)**: Считывает данные из стандартного ввода (stdin), преобразует их в верхний регистр и записывает обратно в стандартный вывод (stdout).
3. **Второй дочерний процесс (child2.c)**: Считывает данные из стандартного ввода (stdin), заменяет пробельные символы на подчеркивания и записывает обратно в стандартный вывод (stdout).

**Код программы**

parent.c

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <stdio.h>

#define BUFFER\_SIZE 256

int main() {

    int pipe1[2], pipe2[2];

    pid\_t child1, child2;

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1)

        exit(EXIT\_FAILURE);

    child1 = fork();

    if (child1 == -1) {

        exit(EXIT\_FAILURE);

    }

    if (child1 == 0) {

        close(pipe1[1]);

        close(pipe2[0]);

        dup2(pipe1[0], STDIN\_FILENO);

        dup2(pipe2[1], STDOUT\_FILENO);

        close(pipe1[0]);

        close(pipe2[1]);

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

        exit(EXIT\_FAILURE);

    }

    close(pipe1[0]);

    close(pipe2[1]);

    char buffer[BUFFER\_SIZE];

    int bytes\_read;

    while ((bytes\_read = read(STDIN\_FILENO, buffer, sizeof(buffer))) > 0) {

        write(pipe1[1], buffer, bytes\_read);

    }

    close(pipe1[1]);

    waitpid(child1, NULL, 0);

    child2 = fork();

    if (child2 == -1) {

        exit(EXIT\_FAILURE);

    }

    if (child2 == 0) {

        close(pipe2[1]);

        dup2(pipe2[0], STDIN\_FILENO);

        dup2(STDOUT\_FILENO, STDOUT\_FILENO);

        close(pipe2[0]);

        execl("./out/child2", "child2", NULL);

        exit(EXIT\_FAILURE);

    }

    close(pipe2[0]);

    waitpid(child2, NULL, 0);

    return 0;

}

child1.c

#include <unistd.h>

#include <ctype.h>

int main() {

    char buffer[256];

    int bytes\_read;

    while ((bytes\_read = read(STDIN\_FILENO, buffer, sizeof(buffer))) > 0) {

        for (int i = 0; i < bytes\_read; i++) {

            buffer[i] = toupper(buffer[i]);

        }

        write(STDOUT\_FILENO, buffer, bytes\_read);

    }

    return 0;

}

child2.c

#include <unistd.h>

#include <ctype.h>

int main() {

    char buffer[256];

    int bytes\_read;

    while ((bytes\_read = read(STDIN\_FILENO, buffer, sizeof(buffer))) > 0) {

        for (int i = 0; i < bytes\_read; i++) {

            if (buffer[i] == ' ') {

                buffer[i] = '\_';

            }

        }

        write(STDOUT\_FILENO, buffer, bytes\_read);

    }

    return 0;

}

**Протокол работы программы**

**Тестирование:**

test.sh

#!/bin/bash

function print\_message {

    echo "--------------------------------------------------"

    echo "$1"

    echo "--------------------------------------------------"

}

print\_message "Start program..."

make

if [ $? -ne 0 ]; then

    echo "Error: make failed!"

    exit 1

fi

input\_file=$(mktemp)

output\_file=$(mktemp)

echo "hello world" > $input\_file

echo "this is a test" >> $input\_file

./out/parent < $input\_file > $output\_file

actual\_output=$(cat $output\_file)

echo "$actual\_output"

rm -f $input\_file $output\_file

make clean

-------------------------------------------------------

**./test.sh**

**--------------------------------------------------**

**Start program...**

**--------------------------------------------------**

**gcc -Wall -Werror -c parent.c -o parent.o**

**gcc -o out/parent parent.o**

**gcc -Wall -Werror -c child1.c -o child1.o**

**gcc -o out/child1 child1.o**

**gcc -Wall -Werror -c child2.c -o child2.o**

**gcc -o out/child2 child2.o**

**HELLO\_WORLD**

**THIS\_IS\_A\_TEST**

**rm -f out/parent out/child1 out/child2 parent.o child1.o child2.o**

Strace:

strace -f -o strace\_output.txt ./out/parent < input\_file > output\_file

1751 execve("./out/parent", ["./out/parent"], 0x7fff64dec2a8 /\* 26 vars \*/) = 0

1751 brk(NULL) = 0x5640f8cf0000

1751 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc3dc82e000

1751 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

1751 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

1751 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=19711, ...}) = 0

1751 mmap(NULL, 19711, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fc3dc829000

1751 close(3) = 0

1751 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1751 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

1751 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

1751 fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

1751 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

1751 mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fc3dc617000

1751 mmap(0x7fc3dc63f000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fc3dc63f000

1751 mmap(0x7fc3dc7c7000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7fc3dc7c7000

1751 mmap(0x7fc3dc816000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7fc3dc816000

1751 mmap(0x7fc3dc81c000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fc3dc81c000

1751 close(3) = 0

1751 mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc3dc614000

1751 arch\_prctl(ARCH\_SET\_FS, 0x7fc3dc614740) = 0

1751 set\_tid\_address(0x7fc3dc614a10) = 1751

1751 set\_robust\_list(0x7fc3dc614a20, 24) = 0

1751 rseq(0x7fc3dc615060, 0x20, 0, 0x53053053) = 0

1751 mprotect(0x7fc3dc816000, 16384, PROT\_READ) = 0

1751 mprotect(0x5640f8189000, 4096, PROT\_READ) = 0

1751 mprotect(0x7fc3dc866000, 8192, PROT\_READ) = 0

1751 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

1751 munmap(0x7fc3dc829000, 19711) = 0

1751 pipe2([3, 4], 0) = 0

1751 pipe2([5, 6], 0) = 0

1751 clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fc3dc614a10) = 1752

1752 set\_robust\_list(0x7fc3dc614a20, 24 <unfinished ...>

1751 close(3) = 0

1752 <... set\_robust\_list resumed>) = 0

1751 close(6 <unfinished ...>

1752 close(4 <unfinished ...>

1751 <... close resumed>) = 0

1752 <... close resumed>) = 0

1751 read(0, <unfinished ...>

1752 close(5 <unfinished ...>

1751 <... read resumed>"hellow world\r\nThis is the test\r\n", 256) = 32

1752 <... close resumed>) = 0

1751 write(4, "hellow world\r\nThis is the test\r\n", 32 <unfinished ...>

1752 dup2(3, 0 <unfinished ...>

1751 <... write resumed>) = 32

1752 <... dup2 resumed>) = 0

1751 read(0, <unfinished ...>

1752 dup2(6, 1 <unfinished ...>

1751 <... read resumed>"", 256) = 0

1752 <... dup2 resumed>) = 1

1751 close(4 <unfinished ...>

1752 close(3 <unfinished ...>

1751 <... close resumed>) = 0

1752 <... close resumed>) = 0

1751 wait4(1752, <unfinished ...>

1752 close(6) = 0

1752 execve("./out/child1", ["child1"], 0x7ffcba53d578 /\* 26 vars \*/) = 0

1752 brk(NULL) = 0x5583afb86000

1752 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f42a74e9000

1752 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

1752 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

1752 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=19711, ...}) = 0

1752 mmap(NULL, 19711, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f42a74e4000

1752 close(3) = 0

1752 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1752 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

1752 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

1752 fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

1752 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

1752 mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f42a72d2000

1752 mmap(0x7f42a72fa000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f42a72fa000

1752 mmap(0x7f42a7482000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7f42a7482000

1752 mmap(0x7f42a74d1000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7f42a74d1000

1752 mmap(0x7f42a74d7000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f42a74d7000

1752 close(3) = 0

1752 mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f42a72cf000

1752 arch\_prctl(ARCH\_SET\_FS, 0x7f42a72cf740) = 0

1752 set\_tid\_address(0x7f42a72cfa10) = 1752

1752 set\_robust\_list(0x7f42a72cfa20, 24) = 0

1752 rseq(0x7f42a72d0060, 0x20, 0, 0x53053053) = 0

1752 mprotect(0x7f42a74d1000, 16384, PROT\_READ) = 0

1752 mprotect(0x5583af1e3000, 4096, PROT\_READ) = 0

1752 mprotect(0x7f42a7521000, 8192, PROT\_READ) = 0

1752 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

1752 munmap(0x7f42a74e4000, 19711) = 0

1752 read(0, "hellow world\r\nThis is the test\r\n", 256) = 32

1752 write(1, "HELLOW WORLD\r\nTHIS IS THE TEST\r\n", 32) = 32

1752 read(0, "", 256) = 0

1752 exit\_group(0) = ?

1752 +++ exited with 0 +++

1751 <... wait4 resumed>NULL, 0, NULL) = 1752

1751 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=1752, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

1751 clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7fc3dc614a10) = 1753

1751 close(5 <unfinished ...>

1753 set\_robust\_list(0x7fc3dc614a20, 24 <unfinished ...>

1751 <... close resumed>) = 0

1753 <... set\_robust\_list resumed>) = 0

1751 wait4(1753, <unfinished ...>

1753 close(6) = -1 EBADF (Bad file descriptor)

1753 dup2(5, 0) = 0

1753 dup2(1, 1) = 1

1753 close(5) = 0

1753 execve("./out/child2", ["child2"], 0x7ffcba53d578 /\* 26 vars \*/) = 0

1753 brk(NULL) = 0x5574e3b1e000

1753 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fb614534000

1753 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

1753 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

1753 fstat(3, {st\_mode=S\_IFREG|0644, st\_size=19711, ...}) = 0

1753 mmap(NULL, 19711, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fb61452f000

1753 close(3) = 0

1753 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1753 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

1753 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

1753 fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

1753 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

1753 mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fb61431d000

1753 mmap(0x7fb614345000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fb614345000

1753 mmap(0x7fb6144cd000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7fb6144cd000

1753 mmap(0x7fb61451c000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7fb61451c000

1753 mmap(0x7fb614522000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fb614522000

1753 close(3) = 0

1753 mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fb61431a000

1753 arch\_prctl(ARCH\_SET\_FS, 0x7fb61431a740) = 0

1753 set\_tid\_address(0x7fb61431aa10) = 1753

1753 set\_robust\_list(0x7fb61431aa20, 24) = 0

1753 rseq(0x7fb61431b060, 0x20, 0, 0x53053053) = 0

1753 mprotect(0x7fb61451c000, 16384, PROT\_READ) = 0

1753 mprotect(0x5574e2bc7000, 4096, PROT\_READ) = 0

1753 mprotect(0x7fb61456c000, 8192, PROT\_READ) = 0

1753 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

1753 munmap(0x7fb61452f000, 19711) = 0

1753 read(0, "HELLOW WORLD\r\nTHIS IS THE TEST\r\n", 256) = 32

1753 write(1, "HELLOW\_WORLD\r\nTHIS\_IS\_THE\_TEST\r\n", 32) = 32

1753 read(0, "", 256) = 0

1753 exit\_group(0) = ?

1753 +++ exited with 0 +++

1751 <... wait4 resumed>NULL, 0, NULL) = 1753

1751 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=1753, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

1751 exit\_group(0) = ?

1751 +++ exited with 0 +++

**Вывод**

В рамках лабораторной работы была разработана программа, демонстрирующая межпроцессное взаимодействие с использованием каналов (pipes) и системных вызовов. Программа состоит из родительского процесса и двух дочерних процессов, которые обрабатывают данные последовательно. Родительский процесс считывает данные из стандартного ввода, передает их первому дочернему процессу, который преобразует их в верхний регистр, затем передает обработанные данные второму дочернему процессу, который заменяет пробельные символы на подчеркивания, и выводит окончательный результат в стандартный вывод.