

Федеральное государственное бюджетное образовательное учреждение высшего образования.
«Национально исследовательский университет «Московский энергетический институт»
Кафедра ВМСС

Лабораторная работа №4

ПРОЦЕССЫ В ОС UNIX

Курс: МУЛЬТИЗАДАЧНЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Группа: А-07м-23

Выполнили: Балашов С.А.,
Кретов Н.В., Рогов Д.Р.

Проверил: Орлов Д.А.

Москва 2023 г.

Вариант 2

Задачи 3, 5, 8

3. Разработайте два процесса, из которых ПРОЦЕСС-ПРЕДОК порождает ПРОЦЕСС-ПОТОМОК и через непоименованный канал взаимодействует с ним, передавая содержимое некоторого файла поблочно. Если предок не может войти в файл, он должен передать сообщение на вывод STDERR. Потомок получает блоки файла и их выводит на экран, используя STDOUT. По окончании файла предок передает потомку сигнал завершения и после окончания работы потомка (о чем получает сигнал) прекращает свою работу.

```
#include <stdio>
#include <stdlib>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <csignal>

#define BLOCK_SIZE 32

void handleSignal(int signum) {
    printf("Signal (%d) from pid=[%d] received\n", signum, getpid());
}

void handleFile(int pipeReadFd, const char* filePath) {
    char buf[BLOCK_SIZE];
    ssize_t nbytes;
    int file;
    if ((file = open(filePath, O_RDONLY)) == -1) {
        fprintf(stderr, "Invalid path to file\n");
        return;
    }
    while ((nbytes = read(file, buf, BLOCK_SIZE)) > 0) {
        write(pipeReadFd, buf, nbytes);
    }
    close(file);
    close(pipeReadFd);
    exit(0);
}

void childTask(int *pipefd)
{
    printf("Child process created\n");
    printf("Pid: %d\n", getpid());
    if (signal(SIGUSR1, handleSignal) == SIG_ERR) {
        printf("Fail to set signal function\n");
    }
    close(pipefd[1]);
    char rbuf[BLOCK_SIZE];
    ssize_t nbytes;
    while ((nbytes = read(pipefd[0], rbuf, BLOCK_SIZE)) > 0) {
        write(1, rbuf, nbytes);
    }
    close(pipefd[0]);
}

void parentTask(int *pipefd, const char* filePath)
{
    printf("Parent process created\n");
    printf("Pid: %d\n", getpid());
    if (signal(SIGCHLD, handleSignal) == SIG_ERR) {
        printf("Failed to set signal function\n");
    }
    close(pipefd[0]);
    handleFile(pipefd[1], filePath);
}

void parentJob(const char* filePath) {
    int pipefd[2];
```

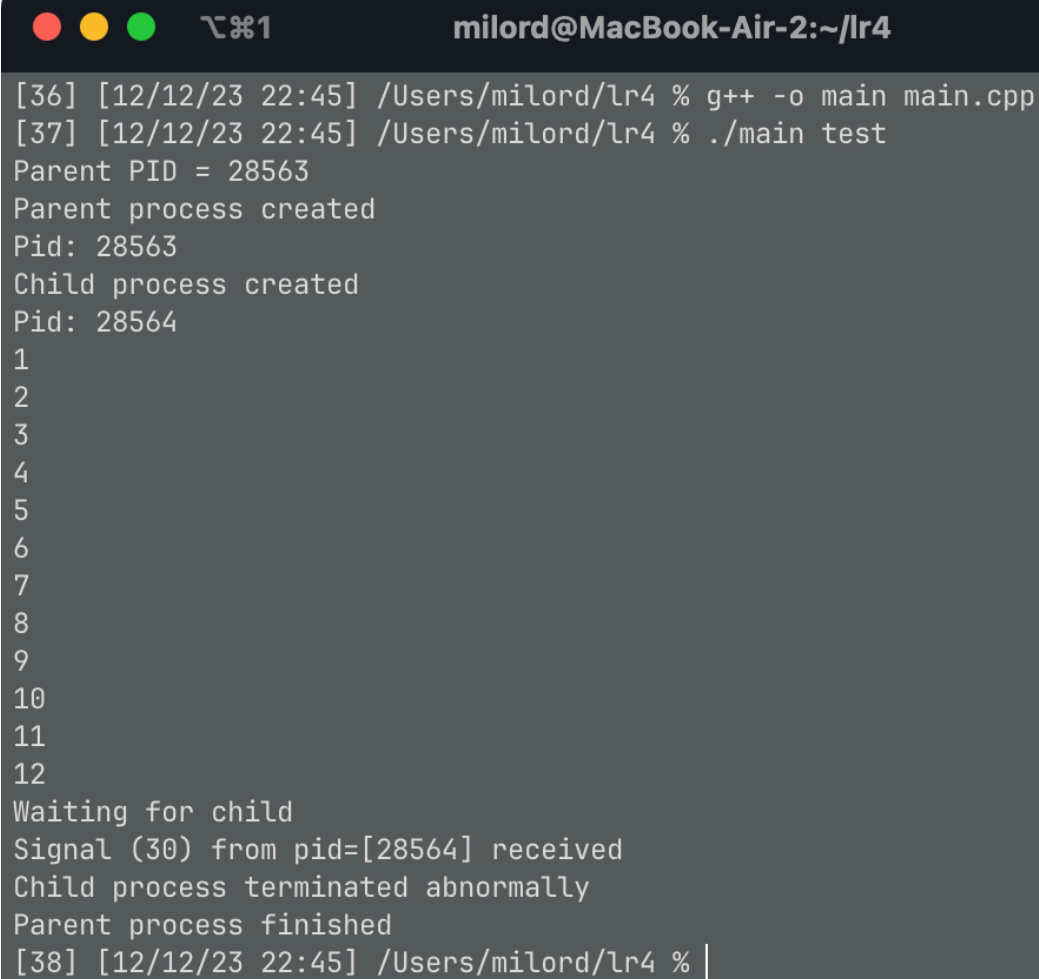
```

pipe(pipefd);
pid_t child_pid;
printf("Parent PID = %d\n", getpid());

if ((child_pid = fork()) == 0) {
    childTask(pipefd);
} else {
    parentTask(pipefd, filePath);
}
printf("\nWaiting for child\n");
kill(child_pid, SIGUSR1);
int child_status;
waitpid(child_pid, &child_status, 0);
if (WIFEXITED(child_status)) {
    int exit_code = WEXITSTATUS(child_status);
    printf("Child process exited with code %d\n", exit_code);
} else {
    printf("Child process terminated abnormally\n");
}
printf("Parent process finished\n");
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Invalid number of arguments\n");
        return 1;
    }
    parentJob(argv[1]);
    return 0;
}

```



```

[36] [12/12/23 22:45] /Users/milord/lr4 % g++ -o main main.cpp
[37] [12/12/23 22:45] /Users/milord/lr4 % ./main test
Parent PID = 28563
Parent process created
Pid: 28563
Child process created
Pid: 28564
1
2
3
4
5
6
7
8
9
10
11
12
Waiting for child
Signal (30) from pid=[28564] received
Child process terminated abnormally
Parent process finished
[38] [12/12/23 22:45] /Users/milord/lr4 % |

```

5. Решите задачу 3 при условии, что потомок снимает информацию из файла, а предок передает ее на экран.

```
#include <stdio>
#include <stdlib>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <csignal>

#define BLOCK_SIZE 32

void handleSignal(int signum) {
    printf("Signal (%d) from pid=%d received\n", signum, getpid());
}

void handleFile(int pipeReadFd, const char* filePath) {
    char buf[BLOCK_SIZE];
    ssize_t nbytes;
    int file;
    if ((file = open(filePath, O_RDONLY)) == -1) {
        fprintf(stderr, "Invalid path to file\n");
        return;
    }
    while ((nbytes = read(file, buf, BLOCK_SIZE)) > 0) {
        write(pipeReadFd, buf, nbytes);
    }
    close(file);
    close(pipeReadFd);
    exit(0);
}

void childTask(int *pipefd, const char* filePath)
{
    printf("Child process created\n");
    printf("Pid: %d\n", getpid());
    if (signal(SIGUSR1, handleSignal) == SIG_ERR) {
        printf("Fail to set signal function\n");
    }
    close(pipefd[0]);
    handleFile(pipefd[1], filePath);
}

void parentTask(int *pipefd)
{
    printf("Parent process created\n");
    printf("Pid: %d\n", getpid());
    if (signal(SIGCHLD, handleSignal) == SIG_ERR) {
        printf("Failed to set signal function\n");
    }
    close(pipefd[1]);
    char rbuf[BLOCK_SIZE];
    ssize_t nbytes;
    while ((nbytes = read(pipefd[0], rbuf, BLOCK_SIZE)) > 0) {
        write(1, rbuf, nbytes);
    }
    close(pipefd[0]);
}

void parentJob(const char* filePath) {
    int pipefd[2];
    pipe(pipefd);
    pid_t child_pid;
    printf("Parent PID = %d\n", getpid());

    if ((child_pid = fork()) == 0) {
        childTask(pipefd, filePath);
    } else {
        parentTask(pipefd);
    }
    printf("\nWaiting for child\n");
    kill(child_pid, SIGUSR1);
}
```

```

int child_status;
waitpid(child_pid, &child_status, 0);
if (WIFEXITED(child_status)) {
    int exit_code = WEXITSTATUS(child_status);
    printf("Child process exited with code %d\n", exit_code);
} else {
    printf("Child process terminated abnormally\n");
}
printf("Parent process finished\n");
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Invalid number of arguments\n");
        return 1;
    }
    parentJob(argv[1]);
    return 0;
}

```

```

[39] [12/12/23 22:45] /Users/milord/lr4 % g++ -o m5 modified5.cpp
[40] [12/12/23 22:46] /Users/milord/lr4 % ./m5 test
Parent PID = 28597
Parent process created
Pid: 28597
Child process created
Pid: 28598
1
2
3
4
5
6
7
8
9
10
11
12
Waiting for child
Signal (20) from pid=[28597] received
Child process exited with code 0
Parent process finished
[41] [12/12/23 22:46] /Users/milord/lr4 % |

```

8. Перекопируйте файл на экран или в другой файл непосредственно через SHELL. Дополнительные условия возьмите из первой задачи, которую будет решать ваша бригада.

modified8.cpp

```

#include <cstdio>
#include <cstdlib>
#include <unistd.h>
#include <fcntl.h>

#define BLOCK_SIZE 32

void handleFile(int pipeReadFd, const char* filePath) {
    char buf[BLOCK_SIZE];
    ssize_t nbytes;
    int file;

```

```

    if ((file = open(filePath, O_RDONLY)) == -1) {
        fprintf(stderr, "Invalid path to file\n");
        return;
    }
    while ((nbytes = read(file, buf, BLOCK_SIZE)) > 0) {
        write(pipeReadFd, buf, nbytes);
    }
    close(file);
    close(pipeReadFd);
    exit(0);
}

int main(int argc, char *argv[])
{
    if (argc != 2) {
        printf("Invalid number of arguments\n");
        return 1;
    }
    handleFile(1, argv[1]);
    return 0;
}

```

modified8-2.cpp

```

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

#define BLOCK_SIZE 32

void handleInput(int pipeWrite)
{
    ssize_t nbytes;
    char Buf[BLOCK_SIZE];
    printf("handler\n");
    while ((nbytes = read(0, Buf, BLOCK_SIZE)) > 0)
    {
        write(pipeWrite, Buf, nbytes);
    }
}

int main()
{
    handleInput(1);
    return 0;
}

```

A terminal window titled "milord@MacBook-Air-2:~/lr4" showing the following commands and output:

```

[22] [12/11/23 17:29] /Users/milord/lr4 % g++ -o mf8 modified8.cpp
[23] [12/11/23 17:30] /Users/milord/lr4 % g++ -o mf82 modified8-2.cpp
[24] [12/11/23 17:30] /Users/milord/lr4 % ./mf8 test | ./mf82
handler
1
2
3
4
5
6
7
8
9
10
11
12%
[25] [12/11/23 17:31] /Users/milord/lr4 % |

```