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-211БВ-24

Студент: Антыгин В.Е.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 13.10.25

Москва, 2025

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:

- fork – создает дочерний процесс.
- pipe – создает канал связи между процессами.
- dup2 – перенаправляет стандартный ввод/вывод.
- execl – запускается другой исполняемый файл внутри текущего процесса, заменяя новым.
- read/write – чтение и запись в файлы.
- close – закрывает файловый дескриптор.
- wait – родительский процесс ожидает завершения дочернего.

Программа реализует конвейерную обработку данных с использованием двух процессов. Родительский процесс считывает строку, введённую пользователем, и передаёт её первому дочернему процессу через пайп. Первый дочерний процесс выполняет преобразование строки, перевод всех символов в верхний регистр, и пересылает результат второму дочернему процессу. Второй дочерний процесс осуществляет дополнительную обработку, замену пробелов символом подчёркивания, после чего отправляет результат обратно родительскому процессу. Родительский процесс получает итоговую строку и выводит её в стандартный поток вывода.

Код программы

parrent.c

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

#define POOL 4096
```

```
int main(int argc, char **argv) {
  if (argc != 3) {
    fprintf(stderr, "Usage: %s <child1> <child2>\n", argv[0]);
    return 3;
  }
  const char *child1Path = argv[1];
  const char *child2Path = argv[2];

  // 0 - read end | 1 - write end
  int pipe1[2], pipe2[2], pipe12[2];
  pipe(pipe1);  // parent to c1
  pipe(pipe2);  // c2 to parent
  pipe(pipe12); // c1 to c2

  if (pipe(pipe1) == -1 || pipe(pipe12) == -1 || pipe(pipe2) == -1) {
    printf("Error occured during opening pipes!\n");
    return 1;
  }

  const pid_t pid1 = fork();
  if (pid1 == 0) {
    // dup2 from fd2 in child to fd  STDIN_FILENO -> pipe12
    dup2(pipe1[0], STDIN_FILENO);
    dup2(pipe12[1], STDOUT_FILENO);

    close(pipe1[1]);
    close(pipe12[0]);
    close(pipe2[0]);
    close(pipe2[1]);

    execl(child1Path, child1Path, NULL);
    printf("Something went wrong during executing child1");
    return 2;
  }

  const pid_t pid2 = fork();
  if (pid2 == 0) {
    dup2(pipe12[0], STDIN_FILENO);
    dup2(pipe2[1], STDOUT_FILENO);

    close(pipe1[0]);
    close(pipe1[1]);
    close(pipe12[1]);
    close(pipe2[0]);

    execl(child2Path, child2Path, NULL);
```

```
    printf("Something went wrong during executing child2");
    return 2;
  }
  close(pipe1[0]);
  close(pipe12[0]);
  close(pipe12[1]);
  close(pipe2[1]);

  char buff[POOL];
  ssize_t nBytes;

  while ((nBytes = read(STDERR_FILENO, buff, sizeof(buff)))) {
    if (nBytes < 0) {
      const char msg[] = "error: failed to read from stdin\n";
      write(STDERR_FILENO, msg, sizeof(msg));
      return 4;
    } else if (buff[0] == '\n') {
      break;
    }

    write(pipe1[1], buff, nBytes);

    nBytes = read(pipe2[0], buff, sizeof(buff));
    write(STDERR_FILENO, buff, nBytes);
  }

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

  wait(NULL);
  wait(NULL);
}
```

child1.c

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

int main() {
  char buf[1024];
  ssize_t n;
  while ((n = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
    for (ssize_t i = 0; i < n; i++)
      buf[i] = toupper(buf[i]);
    write(STDOUT_FILENO, buf, n);
  }
}
```

child2.c

```
#include <unistd.h>

int main() {
  char buf[1024];
  ssize_t n;
  while ((n = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
    for (ssize_t i = 0; i < n; i++)
      if (isspace(buf[i]) && buf[i] != '\n')
        buf[i] = '_';
    write(STDOUT_FILENO, buf, n);
  }
}
```

Протокол работы программы

```
…/build/bin ✕ ./parent.out child1.out child2.out
daskdmnaklsmd qd l      kn lknq  mw;` d '.a.
DASKDMNAKLSMD_QD_L_KN_LKNQ__MW;`_D_'.A.
sald;m a;d m    ; md a
SALD;M_A;D_M_;_MD_A
asdk;m a;sd'    d; mwq1231szsd
ASDK;M_A;SD'_D;_MWQ1231SZSD
m;wmf;dm;fÁSMD:A
M;WMF;DM;FÁSMD:A

______

___

____
                a       .
__A_.
                a       .
_________________A_______.


…/build/bin ❯
```

Вывод

Выполнив эту работу я приобрел навыки работы с созданием процессов, перенаправлением ввода/вывода . При работе столкнулся с недостатком информации, т.к. на лекции с первого раза сложно усвоить абсолютно новый материал, тем более какую-то программу.