Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Управление ресурсами в вычислительных системах

Лабораторная работа №6

Факультет: прикладной математики и информатики

Группа: ПМ-63

Студенты: Кожекин М.В.

Утюганов Д.С.

Преподаватели: Стасышин В. М.

Хайленко Е. А.

Новосибирск

2019

1. Условие (Вариант №6)

Составить программу, моделирующую работу Shell-интерпретатора при обработке командной строки, указанной в варианте. При реализации программы путём выдачи сообщений информировать обо всех этапах ее работы (создан процесс, выполнение команды закончено и т.д.).

ls -lisa | sort | wc -l > a.txt.

1. Анализ задачи

Создаём каналы и порождаем процессы P1 и P2.

Для каждого процесса перенаправляем потоки стандартного ввода / вывода и файла в определённые каналы

**P0:**

exec(ls –lisa)

**P1:**

exec(sort)

**P2:**

exec(wc –l > a.txt)

1. Используемые программные средства

**Функции:**

int fork() – порождение процесса-потомка

int pipe(int[2]) – порождение канала

int close(const char\*file) – закрытие файла или канала

int open(const char\*path, int mode) – открытие файла или именованного канала

int read(int fd, char \*data, int dataSize) – чтение из канала / файла

int waitpid(int pid, int \*status) – ожидание завершения процесса-потомка

int getpid() – определение pid текущего процесса

int exit(int code) – прерывание процесса

1. Спецификация

Программа находится в папке ~/lab6

Чтобы собрать программу нужно ввести “make lab6”

Чтобы запустить программу, нужно использовать команду “./lab6”

В результате работы программы, будет смоделирована работа интерпретатора.

1. Исходный код

**Файл lab6.c**

|  |
| --- |
| // Составить программу, моделирующую работу Shell-интерпретатора при обработке командной строки,  // указанной в варианте. При реализации программы путём выдачи сообщений информировать обо всех  // этапах ее работы (создан процесс, выполнение команды закончено и т.д.). Вариант 6:  // ls -lisa | sort | wc -l > a.txt  #include <signal.h>  #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <unistd.h>  #include <sys/ioctl.h>  #include <fcntl.h>    int main()  {  int fd1[2], fd2[2];  if (pipe(fd1) == -1)  {  printf("Ошибка создания канала №1\n");  exit(-1);  }  if (pipe(fd2) == -1)  {  printf("Ошибка создания канала №2\n");  exit(-1);  }  pid\_t pidP0 = getpid();  printf("P0 (%d):\tСоздание каналов №1 и №2\n", pidP0);  pid\_t pidP1 = fork();  pid\_t pidP2 = 0;  pid\_t pidP;  int returnStatus;  if (pidP1 == 0) {  pidP = 1;  pidP1 = getpid();  pidP2 = fork();  if (pidP2 == 0) {  pidP = 2;  pidP2 = getpid();  }  }  else {  pidP = 0;  }  switch (pidP)  {  // Процесс P0  case 0: {  printf("P0 (%d):\tНачало процесса\n", pidP0);  printf("P0 (%d):\tПроцесс завершён\n", pidP0);  close(fd1[0]);  close(fd2[0]);  close(fd2[1]);  close(1);  dup(fd1[1]);  close(fd1[1]);  if (execl("/bin/ls", "ls", "-lisa", 0) < 0)  exit(-1);  else  exit(0);    }break;  // Процесс P1  case 1: {  printf("P1 (%d):\tНачало процесса\n", pidP1);  printf("P1 (%d):\tЖду завершения процесса P0\n", pidP1);  int status;  waitpid(pidP0, &status, 0);  printf("P1 (%d):\tДождался завершения процесса P0\n", pidP1);  printf("P1 (%d):\tПроцесс завершён\n", pidP1);  close(fd1[1]);  close(0);  dup(fd1[0]);  close(fd1[0]);  close(fd2[0]);  close(1);  dup(fd2[1]);  close(fd2[1]);  if( execl("/bin/sort", "sort", 0) < 0)  exit(-1);  else  exit(0);  }break;  // Процесс P2  case 2: {  printf("P2 (%d):\tНачало процесса\n", pidP2);  printf("P2 (%d):\tЖду завершения процесса P2\n", pidP2);  int status;  waitpid(pidP1, &status, 0);  printf("P2 (%d):\tДождался завершения процесса P1\n", pidP2);  printf("P2 (%d):\tПроцесс завершён\n", pidP2);  close(fd1[0]);  close(fd1[1]);  close(fd2[1]);  close(0);  dup(fd2[0]);  close(fd2[0]);  int fd = open("a.txt", O\_CREAT|O\_RDWR|O\_TRUNC, 0644);  close(1);  dup(fd);  close(fd);  if (execl("/bin/wc", "wc", "-l", 0) < 0)  exit(-1);  else  exit(0);  }break;  }  return 0;  } |

**Содержимое файла Makefile**

|  |
| --- |
| # Makefile for lab #6  lab6:  gcc lab6.c -std=gnu99 -o lab6  clean:  rm lab6  run:  rm lab6 lab6.c; nano lab6.c; gcc lab6.c -std=gnu99 -o lab6; ./lab6; cat a.txt |

1. Результат работы программы

Программа работает должным образом

|  |
| --- |
| -bash-4.2$ ./lab6  P0 (569): Создание каналов №1 и №2  P0 (569): Начало процесса  P0 (569): Процесс завершён  P1 (577): Начало процесса  P1 (577): Жду завершения процесса P0  P1 (577): Дождался завершения процесса P0  P1 (577): Процесс завершён  P2 (584): Начало процесса  P2 (584): Жду завершения процесса P2  P2 (584): Дождался завершения процесса P1  P2 (584): Процесс завершён  -bash-4.2$ cat a.txt  7  -bash-4.2$ ls -lisa | sort | wc -l  7 |