Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Операционные системы»

Студент: Э. Л. Носов
Преподаватель: Е. С. Миронов
Группа: М8О-307Б
Дата:
Оценка:
Подпись:

Москва, 2021

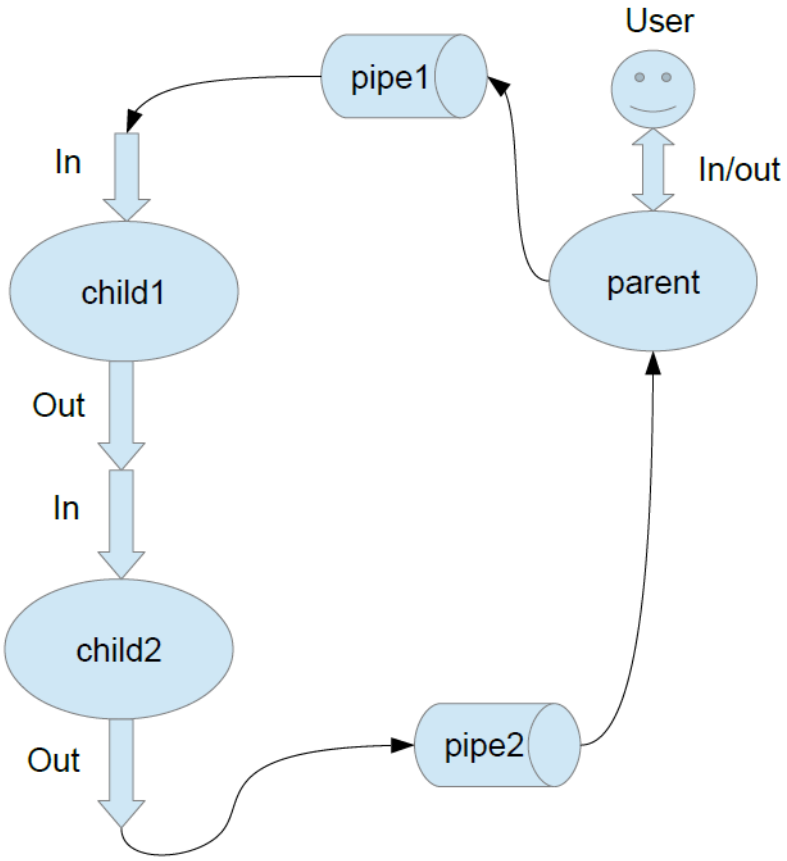
Лабораторная работа №2

Цель работы: Приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

Задание: Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант №12: Child1 переводит строки в верхний регистр. Child2 убирает все задвоенные пробелы.



1 Описание

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.

2 Исходный код

В main-файле при помощи функции pipe(fd) создаются каналы передачи данных, после чего fd[0] указывает на конец для чтения, а fd[1] указывает на конец для записи. При помощи функции fork() создается новый процесс - двойник старого процесса. Функция dup2() изменяет таблицу открытых файлов таким образом, чтобы связать файловые десрипторы стандртных потоков ввода и вывода с нужными концами нужных каналов. Функция execl заменяет код данного процесса на другой код, выполняющий заданную функцию.

main.c

```
#include "unistd.h"
#include "stdio.h"
#include"stdlib.h"

int get_line(char* string, int size){
    char c;
    int n=0;
    while(read(STDIN_FILENO, &c, 1)>0){
        if((c!='\n')&&(n<size)){
        string[n]=c;
        n++;
        }
        else break;
    }
}

int main(){
    const int MAX_CHAR=256;
    int parenttochild1[2], child1tochild2[2], child2toparent[2];
    const char *child1 = "child1", *child2="child2";
    int std_out = fileno(stdout);
  int std_in = fileno(stdin);
  pipe(parenttochild1);
    pipe(child1tochild2);
    pipe(child2toparent);

    int id = fork();
    if(id == -1){
        printf("child1 fork error");
        return -1;
    }else if(id == 0 ){//child1
        close(child2toparent[1]);
        close(child2toparent[0]);
        close(parenttochild1[1]);
    close(child1tochild2[0]);
    if (dup2(parenttochild1[0], std_in) == -1) {
      printf("error copying descriptor to child1's stdin\n");
      return -1;
    }
    if (dup2(child1tochild2[1], std_out) == -1) {
      printf("error copying descriptor to child1's stdout");
      return -1;
    }
    if (execl(child1, child1, NULL) == -1) {
      printf("Failed to exec\n");
      close(parenttochild1[0]);
      close(child1tochild2[1]);
      return -1;
    }
    }
```

```
    else{
        int di=fork();
        if(di == -1){
            printf("child2 fork error");
            return -1;
        }
        else if(di == 0){//child2
            close(parenttochild1[1]);
            close(parenttochild1[0]);
            close(child1tochild2[1]);
        close(child2toparent[0]);
        if (dup2(child1tochild2[0], std_in) == -1) {
          printf("error copying descriptor to child2's stdin\n");
          return -1;
        }
        if (dup2(child2toparent[1], std_out) == -1) {
          printf("error copying descriptor to child2's stdout");
          return -1;
        }
        if (execl(child2, child2, NULL) == -1) {
          printf("Failed to exec\n");
          close(child1tochild2[0]);
          close(child2toparent[1]);
          return -1;
        }
        }
        else{//parent
            close(parenttochild1[0]);
            close(child2toparent[1]);
            char string[MAX_CHAR];
            //printf("here?");
            while(get_line(string, MAX_CHAR)>0){
                //printf("i get here ");
                write(parenttochild1[1], string, MAX_CHAR);
                //printf("and here ");
                read(child2toparent[0], string, MAX_CHAR);
                //printf("but not here");
                printf("%s\n",string);
            }
            //printf("here i come");
        }
    }

  return 0;
}
```

child1.c

```
#include "unistd.h"
#include "stdio.h"
#include"stdlib.h"

int touppercase(char *string, int length){
    for(int i=0; i<length; i++){
        if((string[i]>='a')&&(string[i]<='z')) string[i]+='A'-'a';
        if(string[i]=='\n'){string[i]='\0'; return i;}
    }
}

int main(int argc, char*argv[]){
    printf("child1");
    const int MAX_CHAR= 256;
    char string[MAX_CHAR];
```

```
    while(read(fileno(stdin),string,MAX_CHAR)>0){
        touppercase(string, MAX_CHAR);
        write(fileno(stdout),string, MAX_CHAR);
        //printf("%s\n",string);
    }
    return 0;
}
```

child2.c

```
#include "unistd.h"
#include "stdio.h"
#include"stdlib.h"

void doublespaces(char *string, int length){
    char *str;
    str=malloc(sizeof(char)*length);
    int j=1;
    str[0]=string[0];
    for(int i=1; i<length;i++){
        if((str[j-1]!=' ')&&(string[i]==' ')){
            str[j]=string[i];
            j++;
        }
        else if(string[i]!=' '){
            str[j]=string[i];
            j++;
        }
        else if(string[i]=='\n'){
            str[j]='\0';
            break;
        }
    }
    for(int i=0; i<=j; i++){
        string[i]=str[i];
    }
    free(str);
}

int main(int argc, char*argv[]){
    const int MAX_CHAR = 256;
    printf("child2");
    char string[MAX_CHAR];
    while(read(fileno(stdin),string,MAX_CHAR)>0){
        doublespaces(string, MAX_CHAR);
        write(fileno(stdout),string, MAX_CHAR);
        //printf("%s\n",string);
    }
    return 0;
}
```

3 Консоль

```
maloletniydebil@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OS/labs/lab2$ ./lab2
this is my text
THIS IS MY TEXT
this    is my text
THIS IS MY TEXT
this is       my text
THIS IS MY TEXT
this           is             my          text
THIS IS MY TEXT
```

4 Выводы

Выполнив лабораторную работу №2 по курсу «Операционные системы», я освоил работу с процессами по средствам языка C и системных вызовов pipe(), dup2(), fork() и execl() в операционной системе семейства Unix.