Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Операционные системы»

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

Студент: П. А. Мохляков
Преподаватель: Е. С. Миронов
Группа: М8О-208Б-19
Вариант: 1
Дата:
Оценка:
Подпись:

Москва, 2021

1 Постановка задачи

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.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс принеобходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

Пользователь вводит команды вида: « число число число<endline> ». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int.

2 Сведения о программе

Программа написанна на Си в Unix подобной операционной системе на базе ядра Linux. В программе создается дочерний процесс, в который перенаправляются данные из pipe.

Дочерний прочесс принимает строку чисел и находит их сумму, ответ записывая в файл. Имя файла задается пользователем

Родительский процесс считывает вводные данные у пользовотеля и пердет их дочернему процессу через pipe.

Программа завершает работу при окончании ввода, то есть нажатии CTRL+D.

3 Общий метод и алгоритм решения

При запуске программы прользоваательль может ввести имя файла, который создаст дочерняя программа. Считывание происходит посредством **getline()**.

После запуска создается pipe, два файловых дескриптора которого, записываюся в массив fd из двех элементов.

После этого создается дочерний процесс с помощью **fork()**. В нем дескриптор потока ввода заменяется на поток вывода из pipe. Таким образом, когда родитель запишет что-то в pipe ребенок сможет это считать, как будто ввод происходит из консоли. После замены дескрипторов вызывается дочерняя прогграмма с помощью **execl()**. В нее имя файла передается как параметр при запуске. Дальее пака не стретим конец ввода мы считываем число и символ за ним. Если этот символ пробел, то считанное число просто прибавляется к сумме, если символ является символом конца строки, сумма записывается в файл, а сумма обнуляется.

Тем временем родитель посимвольно считывает данные от пользователя и записывает их в файл. При нажатии CTRL+D пользователь сигнализирует о конце ввода. Родительский процесс завершает работу, а вместе с ним и дочерний.

4 Листинг программы

main.c

```
1  #include "unistd.h"
2  #include "string.h"
3  #include "stdio.h"
4  int main(){
5      int fd[2];
6      if(pipe(fd) < 0){
7          printf("Error pipe create\n");
8          return -1;
9      }
10     char *filename = NULL;
11     size_t sizename = 0;
12     getline(&filename,&sizename,stdin);
13     filename[strlen(filename)-1] = '\0';
14     int id = fork();
15     if(id == -1){
16         printf("Error fork\n");
17         return -1;
18     }else if(id == 0){
19         close(fd[1]);
20         dup2(fd[0],0);
21         execl("./child","child",filename,(char*) NULL);
22     } else {
23         close(fd[0]);
24         char ch;
25         while(scanf("%c",&ch) != EOF){
26             write(fd[1],&ch,sizeof(ch));
27         }
```

```
28          close(fd[1]);
29      }
30      return 0;
31  }
```

sum.c

```
1   #include "stdio.h"
2   #include "string.h"
3
4   int main(int argc, char **argv){
5       if(argc != 2){
6           return -1;
7       }
8       long long sum = 0;
9       int num;
10      char ch;
11      FILE *file;
12      file = fopen(argv[1], "w");
13      while(scanf("%d%c",&num,&ch) != EOF){
14          sum += num;
15          if(ch == '\n'){
16              fprintf(file,"%lld\n",sum);
17              sum = 0;
18          }
19      }
20      fclose(file);
21      return 0;
22  }
```

5 Демонстрация работы программы

```
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB2$ gcc main.c -o parent
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB2$ gcc sum.c -o child
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB2$ ./parent
test
1 2 3 4 5
0 0 0
12 45 34 54
24 -5
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB2$ cat test
15
0
145
19
```

```
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB2$ strace -f -e
trace="read,write,dup2,pipe" -o log.txt ./parent
test2
1 2 3
0 0
2 -1
pavel@DESKTOP-K5KMLPV:~/Project/mai/2_course/OS/LB2$ cat log.txt
2452  read(3,"\177ELF\\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q
\2\0\0\0\0\0"...,832) = 832
2452  pipe([3,4])                       = 0
2452  read(0,"test2\ n",1024)           = 6
2452  read(0, <unfinished ...>
2454  dup2(3,0)                         = 0
2454  read(4,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q
\2\0\0\0\0\0"...,832) = 832
2454  read(0, <unfinished ...>
2452  <... read resumed>"1 2 3\n",1024) = 6
2452  write(4,"1",1)                    = 1
2454  <... read resumed>"1",4096)       = 1
2452  write(4," ",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)               = 1
2454  <... read resumed>" ",4096)       = 1
2452  write(4,"2",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)               = 1
2454  <... read resumed>"2",4096)       = 1
2452  write(4," ",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)               = 1
2454  <... read resumed>" ",4096)       = 1
2452  write(4,"3",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)               = 1
2454  <... read resumed>"3",4096)       = 1
2452  write(4,"\n",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)               = 1
2454  <... read resumed>"\n",4096)      = 1
2452  read(0, <unfinished ...>
2454  read(0, <unfinished ...>
```

```
2452  <... read resumed>"0 0\n",1024)  = 4
2452  write(4,"0",1)                   = 1
2454  <... read resumed>"0",4096)      = 1
2452  write(4," ",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)              = 1
2454  <... read resumed>" ",4096)      = 1
2452  write(4,"0",1)                   = 1
2454  read(0, <unfinished ...>
2452  write(4,"\n",1)                  = 1
2454  <... read resumed>"0\n",4096)    = 2
2452  read(0, <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... read resumed>"2 -1\n",1024) = 5
2452  write(4,"2",1)                   = 1
2454  <... read resumed>"2",4096)      = 1
2452  write(4," ",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)              = 1
2454  <... read resumed>" ",4096)      = 1
2452  write(4,"-",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)              = 1
2454  <... read resumed>"-",4096)      = 1
2452  write(4,"1",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)              = 1
2454  <... read resumed>"1",4096)      = 1
2452  write(4,"\n",1 <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... write resumed>)              = 1
2454  <... read resumed>"\n",4096)     = 1
2452  read(0, <unfinished ...>
2454  read(0, <unfinished ...>
2452  <... read resumed>"",1024)       = 0
2454  <... read resumed>"",4096)       = 0
2454  write(4,"6\n0\n1\n",6)           = 6
2452  +++ exited with 0 +++
2454  +++ exited with 0 +++
```

6 Вывод

Одна из основных задач операционной системы - это управление процессами. В большинстве случаев она сама создает процессы для себя и при запуске других программ. Тем не менее бывают случаи, когда необходимо создавать процессы вручную.

В языке Си есть функционал, который позволит нам внутри нашей программы создать дополнительный, дочерний процесс. Этот процесс будет работать параллельно с родительским.

Для этого в языке Си на Unix-подобных ОС используется библеотека unistd.h. Эта библеотека позволяет совершать системные вызовы, которые связаны с вводом/выводом, управлением файлами, каталогами и работой с процессами и запуском программ. Для создания дочерних процессов используется функция fork. При этом с помощью ветвлений в коде можно отделить код родителя от ребенка. У ребенка при этом можно заменить программу, испрользуя для этого функцию exec, а обеспечить связь с помощью pipe.

Подобный функционал есть во многих языках программирования, так как большинство современных программ состаят более, чем из одного процесса.