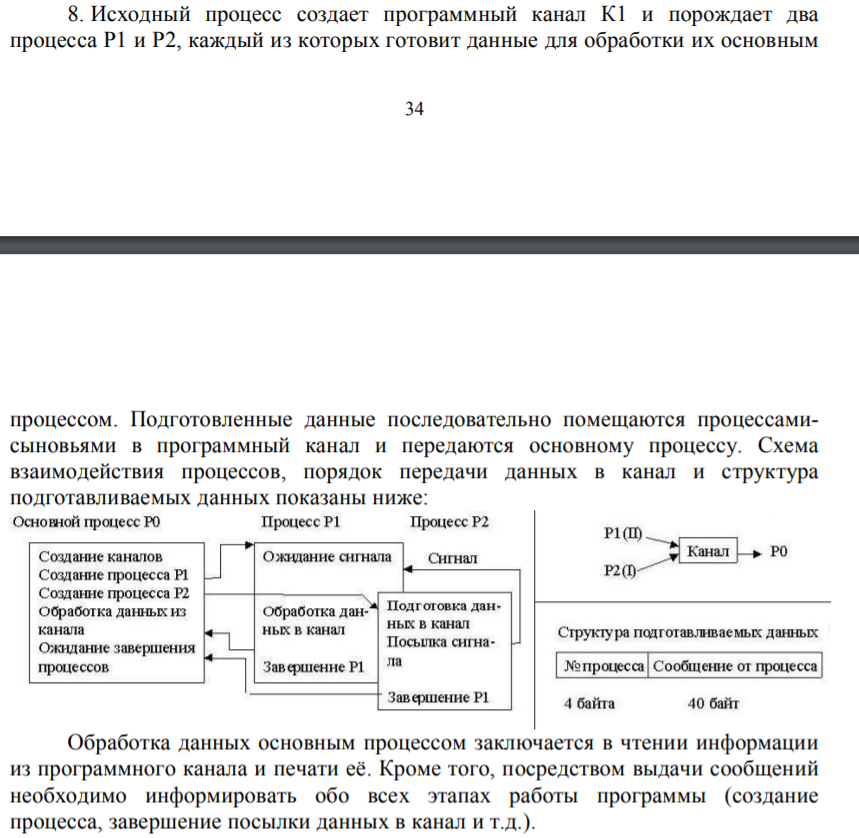
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
| C:\Users\kimev\AppData\Local\Temp\7zO0E6F35D7\4 ЦВЕТ (1).png | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Практическое задание № 5 | | |
| по дисциплине «Управление ресурсами в вычислительных системах» | | |
|  | | |
| Место для ввода текста. | | |
| **СИНХРОНИЗАЦИЯ ПРОЦЕССОВ** | | |
|  | Факультет: | ПМИ |
| Группа: | ПМ-71 |
|  | Место для ввода текста. |
| Студент: | Антонов Сергей |
|  |  |
| Вариант: | 8 |
| Преподаватели: | Стасышин В. М. |
|  | Сивак М. А. |
|  | | |
| Новосибирск | | |
| 2020 | | |

1. **Цель**

Практическое освоение механизма синхронизации процессов и их  
взаимодействия посредством программных каналов.

1. **Задание**



1. **Алгоритм решения**
2. **Основной процесс**
   1. Создать программный канал.
   2. Породить процесс P1.
   3. Породить процесс P2.
   4. По мере поступления данных выводить их на экран.
   5. Дождаться завершения процессов.
   6. Закрыть канал.
3. **Процесс P1**
   1. Ждать сигнала от процесса P2.
   2. Вывести в канал PID процесса.
   3. Вывести в канал целые числа от 1 до того числа, на котором наступит предел 40 байт.
4. **Процесс P2**
   1. Вывести в канал PID процесса.
   2. Вывести в канал вещественные числа от 1 до того числа, на котором наступит предел 40 байт.
   3. Послать сигнал процессу P1.
5. **Программные компоненты**

* Для порождения нового процесса (процесс-потомок) используется системный вызов fork();
* Для посылки сигнала используется системный вызов kill();
* Программный канал создается с помощью системного вызова pipe();
* Все операции передачи данных через канал выполняются с помощью системных вызовов ввода-вывода read/write;
* Открытие файла происходит с помощью функции open();
* exit() – выход из программы;

1. **Спецификация программы**

gcc -o laba5 laba5.c

Программа находится в корневом каталоге pmi-b7108@students/laba5 ~, исходный код в файле laba5.с. Исполняемый файл - ./laba5

1. **Текст программы**

#include <sys/types.h>

#include <sys/unistd.h>

#include <sys/signal.h>

#include <sys/wait.h>

#include <stdio.h>

#include <stdlib.h>

/\*Вывод ошибок\*/

void Close(int fd)

{

if (close(fd) == -1)

{

perror("Error close");

exit(1);

}

}

void Error(int status[2])

{

if (status[0] && WIFSIGNALED(status[0]))

{

fprintf(stderr, "Child #P1 received unexpected signal %i\n", WTERMSIG(status[0]));

exit(1);

}

if (status[1] && WIFSIGNALED(status[1]))

{

fprintf(stderr, "Child #P2 received unexpected signal %i\n", WTERMSIG(status[1]));

exit(1);

}

if (status[0] && WIFSTOPPED(status[0]) && WSTOPSIG(status[0]))

{

fprintf(stderr, "Child #P1 stopped with signal %i\n", WSTOPSIG(status[0]));

exit(1);

}

if (status[1] && WIFSTOPPED(status[1]) && WSTOPSIG(status[1]))

{

fprintf(stderr, "Child #P2 stopped with signal %i\n", WSTOPSIG(status[1]));

exit(1);

}

}

void ignor(int i) { }

int main()

{

int fifo[2];

double data[2];

int status[2];

int flag, i;

/\* Создаем канал \*/

if (pipe(fifo) == -1)

{

fprintf(stderr, "Cant create pipe!\n");

exit(1);

}

printf("P0 start\nCreate canal\n");

signal(SIGUSR1, ignor);

/\* Создаем процесс #P1 \*/

pid\_t child1 = fork();

if (child1 == -1)

{

fprintf(stderr, "Cant fork child #P1!\n");

Close(fifo[0]);

Close(fifo[1]);

exit(1);

}

else if (!child1)

{

printf("P0 create P1\nP0 waiting P1\n");

signal(SIGUSR1, ignor);

pid\_t child\_p1 = getpid();

printf("Child #P1: %i created.\n", child\_p1);

printf("P1 waiting P2\n");

for (flag = 0; flag < 10; flag++)

if (kill(getppid(), SIGUSR1) != -1)

break;

if (flag == 10)

{

perror("Signal send error \n");

return 0;

}

sigpause(SIGUSR1);

printf("Child #P1: signal accepted.\n");

/\* Счетчик для 40-битов\*/

int n = 40 / sizeof(double);

for (i = 0; i < n; i++)

{

data[0] = (double)(i + 1);

if (write(fifo[1], &child\_p1, sizeof(int)) == -1)

{

perror("Error write");

exit(1);

}

if (write(fifo[1], &data[0], sizeof(double)) == -1)

{

perror("Error write");

exit(1);

}

}

printf("Child #P1: work done.\n");

exit(0);

}

sigpause(SIGUSR1);

/\*\* Создаем #P2 \*/

pid\_t child2 = fork();

if (child2 == -1)

{

fprintf(stderr, "Can`t fork child #P2!\n");

Close(fifo[0]);

Close(fifo[1]);

for (flag = 0; flag < 10; flag++)

if (kill(child1, SIGKILL) != -1)

break;

if (flag == 10)

{

perror("Signal send error \n");

return 0;

}

exit(1);

}

if (!child2)

{

pid\_t child\_p2 = getpid();

printf("Child #P2: %i created.\n", child\_p2);

/\* Счетчик для 40-битов\*/

int n = 40 / sizeof(double);

for (i = 0; i < n; i++)

{

data[1] = (double)(i + 1);

if (write(fifo[1], &child\_p2, sizeof(int)) == -1)

{

perror("Error write");

exit(1);

}

if (write(fifo[1], &data[1], sizeof(double)) == -1)

{

perror("Error write");

exit(1);

}

}

for (flag = 0; flag < 10; flag++)

if (kill(child1, SIGUSR1) != -1)

break;

if (flag == 10)

{

perror("Signal send error \n");

return 0;

}

printf("Child #P2: signal sended.\n");

printf("Child #P2: work done.\n");

exit(0);

}

Close(fifo[1]);

/\* Вывод результата \*/

pid\_t pid;

while (read(fifo[0], &pid, sizeof(int)))

{

if (pid == child1)

{

if (read(fifo[0], &data[0], sizeof(double)) == -1)

{

perror("Error read");

exit(1);

}

printf("%i: %lf\n", pid, data[0]);

}

else if (pid == child2)

{

if (read(fifo[0], &data[1], sizeof(double)) == -1)

{

perror("Error read");

exit(1);

}

printf("%i: %lf\n", pid, data[1]);

}

else

{

fprintf(stderr, "Unknown PID value in pipe!\n");

for (flag = 0; flag < 10; flag++)

if (kill(child1, SIGKILL) != -1)

break;

if (flag == 10)

{

perror("Signal send error \n");

return 0;

}

for (flag = 0; flag < 10; flag++)

if (kill(child2, SIGKILL) != -1)

break;

if (flag == 10)

{

perror("Signal send error \n");

return 0;

}

}

}

/\*\* Ждем остановки дочерних процессов \*/

waitpid(child1, &status[0], WUNTRACED);

waitpid(child2, &status[1], WUNTRACED);

Close(fifo[0]);

printf("P0 get FinishSignal by P1\n");

printf("P0 finish\n");

Error(status);

return 0;

}

1. **Тестирование программы**

