Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра теоретической и прикладной информатики

Лабораторная работа №2 по дисциплине

«Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМИ-12

Студенты: Швадченко А. В.

Михайловский М. А.

Вариант: 4

Преподаватели: Стасышин В. М., Сивак М. А.

Новосибирск

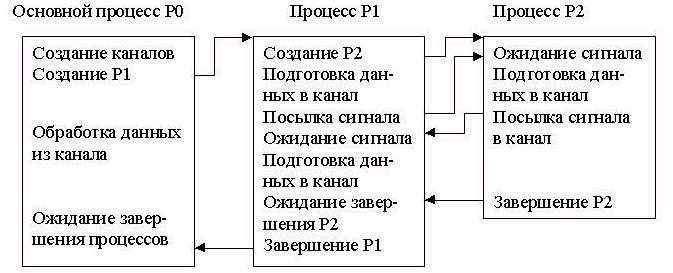
2024

1. ***Цель работы***

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

1. ***Задание***

Исходный процесс создает программный канал К1 и порождает новый процесс Р1, а тот, в свою очередь, еще один процесс Р2, каждый из которых готовит данные для обработки их основным процессом. Подготовленные данные последовательно помещаются процессами-сыновьями в программный канал и передаются основному процессу. Схема взаимодействия процессов, порядок передачи данных в канал и структура подготавливаемых данных показаны ниже:



Обработка данных основным процессом заключается в чтении информации из программного канала и печати её. Кроме того, посредством выдачи сообщений необходимо информировать обо всех этапах работы программы (создание процесса, завершение посылки данных в канал и т.д.).

1. ***Описание используемых структур***

Создание, завершение процесса, получение информации о процессе, - fork(), exit(), getpid(), getppid();

Синхронизация процессов - signal(), kill(), wait(), pause();

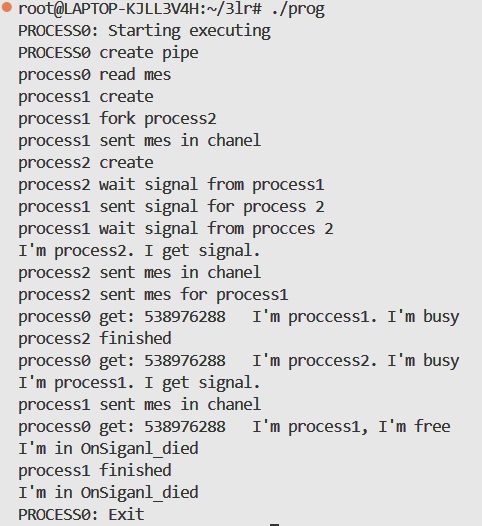
Создание информационного канала и работа с ним - pipe(), read(), write().

1. ***Спецификация***

Программа находится в каталоге /root/3lr и называется prog.c.

Чтобы получить исполняемый файл необходимо выполнить команду: gcc –o [имя\_исполняемого\_файла] prog.c. Далее можно запустить исполняемый файл командой: ./[ имя\_исполняемого\_файла].

1. ***Тесты***



1. ***Текст программы***

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/dir.h>

#include<fcntl.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<signal.h>

#include <locale.h>

void Onsignal0()

{

printf("I'm process0. I get signal.\n");

}

void Onsignal1()

{

printf("I'm process1. I get signal.\n");

}

void Onsignal2()

{

printf("I'm process2. I get signal.\n");

}

void OnSignal\_died()

{

printf("I'm in OnSiganl\_died\n");

}

int main()

{

setlocale (LC\_TIME, "ru\_RU");

int i, n, pd[2], pid1, pid2, ppid, pid, p; //переменные для цикла, параметр и для каналов

char message[500];

int status;

long int gp;

printf("PROCESS0: Starting executing\n");

printf("PROCESS0 create pipe\n");

//Создаём канал

p = pipe(pd);

if (p == -1)

{

printf("ERROR!!!"); //Ошибка при создании канала

return 0;

};

//устанавливаем обработчики сигнала

signal(SIGUSR1, Onsignal0);

signal(SIGCHLD, OnSignal\_died);

pid = fork();//создаем процесс 1

switch(pid)

{

case -1:

{

printf("error! process1 not create\n");

return 0;

}

case 0:

{

printf("process1 create\n");

printf("process1 fork process2\n");

pid = fork(); // процесс 2

//устанавливаем обработчики сигнала

signal(SIGUSR1, Onsignal1);

signal(SIGCHLD, OnSignal\_died);

switch(pid)

{

case -1:

{

printf("error! process2 not create\n");

return 0;

}

case 0:

{

printf("process2 create\n");

printf("process2 wait signal from process1\n");

//устанавливаем обработчики сигнала

signal(SIGUSR1, Onsignal2);

signal(SIGCHLD, OnSignal\_died);

pause();//ждем сигнала

gp = getpid();//получаем идентификатор процесса

strcpy(message, " I'm proccess2. I'm busy\n");

printf("process2 sent mes in chanel\n");

//записываем в канал информацию от второго процесса

write(pd[1], gp, sizeof(gp));

write(pd[1], message, sizeof(message));

ppid = getppid();//получаем предка второго процесса

printf("process2 sent mes for process1\n");

kill(ppid, SIGUSR1);//отправляем сигнал процессу 1

printf("process2 finished \n");

exit(0);

}

default:

{

//процесс потомка

pid2 = pid;

gp = getpid(); //получаем идентификатор

printf("process1 sent mes in chanel\n");

strcpy(message, " I'm proccess1. I'm busy\n");

//записываем в канал информацию от первого процесса

write(pd[1], gp, sizeof(gp));

write(pd[1], message, sizeof(message));

printf("process1 sent signal for process 2\n");

kill(pid2, SIGUSR1);//отправляем сигнал второму

//потомок P1 приостанавливает свое выполнение до поучения сигнала SIGUSR1

printf("process1 wait signal from procces 2\n");

pause(); //ждем сигнала

printf("process1 sent mes in chanel\n");

strcpy(message, " I'm process1, I'm free\n");

//записываем в канал информацию от первого процесса

write(pd[1],gp,sizeof(gp));

write(pd[1], message, sizeof(message));

//ожидать завершения работы потомка

wait(&status);

printf("process1 finished\n");

exit(0);

}

}

default:

{

//родительский процесс

pid1 = pid;

printf("process0 read mes\n");

for(i=0;i<3;i++)

{

// Читаем из канала информацию подготовленную потомками

read(pd[0],&gp,sizeof(gp));

read(pd[0], &message, sizeof(message));

// Получаем номер процесса записавшего эти данные в канал

pid = gp;

printf("process0 get: %d\t%s",pid, message);

}

}

}

}

//ожидать завершения работы потомка

wait();

printf("PROCESS0: Exit\n");

}