ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА (САМАРСКИЙ УНИВЕРСИТЕТ)»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ
Кафедра программных систем

А.В. Баландин

Лабораторная работа

Запуск и организация взаимодействия параллельных процессов

Методические указания

Самара
2022

Цель работы

Цель работы - изучить и практически освоить функции операционной системы QNX для запуска параллельных процессов и организации межпроцессного взаимодействия с помощью механизма обмена сообщениями.

Порядок выполнения работы

1. Перед выполнением полученного задания необходимо внимательно изучить следующие стандартные функции операционной системы QNX для запуска и организации взаимодействия параллельных процессов:
   - семейство функций spawn();
   - флаги запуска P_WAIT, P_NOWAIT, P_NOWAITO, P_OVERLAY;
   - функции управления каналом ChannelCreate(), ChannelDestroy(), ConnectAttach(),ConnectDetach();
   - функции передачи сообщений MsgSend(), MsgReceive(), MsgReply(), MsgRead(),MsgWrite().
2. Написать программу, реализующую полученный вариант задания, используя приведённые выше функции. Текст программы должен содержать комментарии выполняемых действиям.

Порядок и форма отчётности

Отчёт по результатам выполнения лабораторной работы включает в себя:

1. Демонстрацию работы многопроцессного приложения в соответствии с заданием.
2. Проверку теоретических знаний по запуску процессов и организации их взаимодействия.
3. Представление оформленного отчёта, по результатам выполнения лабораторной работы.

По структуре отчёт должен содержать:

- стандартно оформленный титульный лист (см. Приложение),
- текст задания с номером задания и номером варианта, краткое описание порядка выполнения приложения,
- исходные тексты программных модулей с комментариями.

Отчёт печатается на бумаге формата А4 через один интервал размер 12пт.

Варианты заданий

ВАРИАНТ №1

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства spawn*(), запускает процессы P2(M2) и P3(M3), передавая им в качестве параметра chid созданного канала, затем переходит в состояние ожидания сообщений по созданному каналу.

Процесс P2 создает свой канал, устанавливает соединение с каналом процесса P1, отправляет ему сообщение о chid своего канала и, получив ответ от P1, переходит в состояние приема сообщений по созданному каналу.

Процесс P3 устанавливает соединение с каналом процесса P1 и посылает ему запрос на получение pid процесса P2 и chid его канала. Получив от P1 ответ (pid и chid), присоединяется к каналу процесса P2 и посылает ему сообщение "P3 send message to P2".

Процесс P2, приняв сообщение от процесса P3, добавляет к нему информацию "P2 send message to P1" и отправляет сформированное таким образом сообщение процессу P1.

Процесс P1, получив сообщение от P2, выдает его на экран терминала, посылает ответ "P1 OK" процессу P2 и терминируется.

Процесс P2, получив ответ от P1, выдает его на экран терминала, посылает ответ "P2 OK" процессу P3 и терминируется.

Процесс P3, получив ответ от P2, выдает его на экран терминала после чего выдает на терминал "P3 OK" и терминируется.

ВАРИАНТ №2

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства spawn*(), запускает процесс P2(M2), передавая ему в качестве параметра chid созданного канала, затем переходит в состояние приема сообщений по своему каналу.

Процесс P2 создает свой канал и, используя функцию семейства spawn*(), запускает процесс P3(M3), передавая ему в качестве параметра chid созданного канала, затем устанавливает соединение с каналом процесса P1, отправляет ему сообщение о pid процесса P3 и переходит в состояние приёма сообщений по созданному каналу.

Процесс P3 создает свой канал, устанавливает соединение с каналом процесса P2 и посылает ему сообщение "P3 loaded". Получив ответ, посылает ему chid своего канала и переходит в состояние приема сообщений по своему каналу.

Процесс P2, приняв первое сообщение от процесса P3, отправляет его процессу P1, получив ответ от P1, принимает chid от процесса P3, посылает ему ответ и передает chid процессу P1. Далее выдает на терминал "P2 loaded" и переходит в состояние приема сообщений по своему каналу.

Процесс P1, получив первое сообщение от P2, выдает его на экран терминала, посылает ответ процессу P2 и принимает второе сообщение, устанавливает соединение с каналом процесса P3 и посылает по нему сообщение "stop", после ответа переходит в состояние приема сообщений по своему каналу.

Процесс P3, получив "stop" от процесса P1, отправляет его процессу P2, печатает "P3 stop" и терминируется.

Процесс P2, получив "stop", отправляет его процессу P1, печатает "P2 stop" и терминируется.

Процесс P1, получив "stop", печатает "P1 stop" и терминируется.

ВАРИАНТ №3

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства spawn*(), запускает процессы P2(M2) и P3(M3), передавая им в качестве параметра chid созданного канала, затем переходит в ожидание сообщений по своему каналу.

Процесс P2 создает свой канал, устанавливает соединение с каналом процесса P1, отправляет ему сообщение о chid своего канала и переходит в состояние приема сообщений по созданному каналу.

Процесс P3 создает свой канал, устанавливает соединение с каналом процесса P1, отправляет ему сообщение о chid своего канала и переходит в состояние приема сообщений по созданному каналу.

Процесс P1, приняв сообщение от процесса P2 или P3 о chid канала, устанавливает соединение и посылает по нему - "P1 send message to P?" (? – номер соответствующего процесса), принимает ответ и выдает его на терминал. После такого взаимодействия с P2 и P3 процесс P1 терминируется.

Процесс P?, получив сообщение от P1, выдает его на экран терминала, посылает ответ "P? OK" процессу P1 и терминируется.

ВАРИАНТ №4

Разработать приложение, состоящее из пяти взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 используя функцию семейства spawn*(), запускает процесс P2(M2) с флагом P_NOWAIT и P3(M2) с флагом P_WAIT.

Процесс P2(M2) создаёт свой канал и, используя функцию семейства spawn*(), запускает процесс P4(M3), передавая в качестве аргумента chid своего канала, затем переходит в состояние приёма сообщений по созданному каналу.

Процесс P3(M2) создаёт свой канал и, используя функцию семейства spawn*(), запускает процесс P5(M3), передавая в качестве аргумента chid своего канала, затем переходит в состояние приёма сообщений по созданному каналу.

Процесс P?(M3) (? – номер соответствующего процесса) устанавливает соединение с каналом родительского процесса P?(M2) и посылает сообщение "P?-OK", получив ответ , выдаёт его на терминал и терминируется.

Процесс P?(M2), получив сообщение от P?(M3), выдает его на экран терминала, посылает ответ "P? OK" процессу P?(M3) и терминируется.

Процесс P1, возобновив свою работу, выдает на терминал сообщение "STOP" и терминируется.

ВАРИАНТ №5

Разработать приложение, которое строится в виде цепочки процессов. Требуется написать два программных модуля – M1, M2. На базе модуля M1 из shell запускается стартовый процесс P1(M1), которому передается в качестве параметра длина цепочки N (количество процессов).

Процесс P1 создает свой канал и, используя функцию семейства spawn*(), запускает процесс P2(M2), передавая в качестве аргумента длину цепочки N и chid своего канала, затем переходит в ожидание прихода сообщений по созданному каналу. После получения сообщения посылает ответ и терминируется.

Процесс P2(M2) устанавливает соединение с каналом родительского процесса и, если N не равно 0, создает свой канал и, используя функцию семейства spawn*(), запускает следующий процесс P?(M2) (? – номер соответствующего процесса) , передавая ему в качестве аргумента N-1 и chid своего канала, затем переходит в состояние приема сообщений по своему каналу. После получения сообщения посылает ответ. Если N=0, то процесс P?(M2) посылает родительскому процессу сообщение "P? OK" и, получив ответ, выдает его на экран и терминируется. Аналогичным образом ведет себя каждый запущенный процесс цепочки.

ВАРИАНТ №6

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства spawn*(), запускает процесс P2(M2), передавая ему в качестве параметра chid созданного канала, затем переходит в состояние приема сообщений по созданному каналу.

Процесс P2 создает свой канал, и, используя функцию семейства spawn*(), запускает процесс P3(M3), передавая ему в качестве параметра chid созданного канала, затем устанавливает соединение с каналом процесса P1 и переходит в состояние приема сообщений по своему каналу.

Процесс P3 устанавливает соединение с каналом процесса P2 и посылает ему запрос на получение pid процесса P1 и chid его канала. Получив от P2 ответ (pid и chid), устанавливает соединение с каналом процесса P1 и посылает ему сообщение "P3 send message to P1". После получения ответа посылает процессу P2 сообщение "P3 stop", затем, получив ответ, выводит на экран "P3 OK" и терминируется.

Процесс P1, получив сообщение от процесса P3, выдает его на терминал, затем посылает ответ и переходит в ожидание сообщений по своему каналу.

Процесс P2, получив сообщение от процесса P3, выводит его на экран и посылает ответ. Затем посылает процессу P1 сообщение "P2 stop", получает ответ, выводит на экран "P2 OK" и терминируется.

Процесс P1, получив сообщение от процесса P2 выдает его на терминал, посылает ответ, выводит на экран "P1 OK" и терминируется.

ВАРИАНТ №7

Разработать приложение, состоящее из пяти взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создаёт свой канал и, используя функцию семейства spawn*(), запускает процессы P2(M2) и P3(M2), передавая им в качестве аргумента chid своего канала, затем переходит в состояние приёма сообщений по своему каналу.

Процесс P2 создаёт свой канал и, используя функцию семейства spawn*(), запускает процесс P4(M3), передавая в качестве аргумента chid своего канала, затем переходит в состояние приёма сообщений по созданному каналу.

Процесс P3 создаёт свой канал и, используя функцию семейства spawn*(), запускает процесс P5(M3), передавая в качестве аргумента chid своего канала, затем переходит в состояние приёма сообщений по созданному каналу.

Процесс P?(M3) (? – номер соответствующего процесса) устанавливает соединение с каналом родительского процесса P?(M2) и посылает запрос на получение pid процесса P1 и chid его канала, затем, после получения ответа (pid и

chid), устанавливает соединение с каналом процесса P1 и посылает ему сообщение "P? loaded". После получения ответа выводит на экран "P? OK" и терминируется.

Процесс P?(M2) после ответа на запрос процесса P?(M3) выводит на терминал "P? OK" и терминируется.

Процесс P1, получив сообщение от процесса P1 или P5, выводит его на экран и посылает ответ. После взаимодействия с P1 и P5 процесс P1 выводит на экран "P1 OK" и терминируется.

ВАРИАНТ №8

Разработать приложение, состоящее из четырёх взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создаёт канал и, используя функцию семейства spawn*(), запускает процесс P2(M2), передавая ему в качестве параметра chid созданного канала, затем переходит в ожидание сообщений по своему каналу.

Процесс P2 создаёт свой канал, и, используя функцию семейства spawn*(), запускает процессы P3(M3) и P4(M3), передавая им в качестве параметра chid своего созданного канала, затем устанавливает соединение с каналом процесса P1 и переходит в состояние приема сообщений по своему каналу.

Процессы P3 и P4 устанавливают соединение с каналом родительского процесса P2, отправляют ему сообщение "P? loaded" (? – номер соответствующего процесса), получив ответ от процесса P2, выводят его на экран, завершают работу (аннулируют созданный канал и соединение) и терминируются.

Процесс P2, получив сообщение от процесса P? (P3 или P1), выводит его на экран и посылает ответ - "P? stop". После такого взаимодействия и с P3 и с P4 процесс P2 посылает процессу P1 сообщение "P2 loaded and executed", после получения ответа от P1 выводит его на экран, завершает работу (аннулирует созданный канал и соединение) и терминируется.

Процесс P1, приняв сообщение от процесса P2, посылает ответ "P2 stop", затем выводит на экран "P1 executed " и терминируется.

ВАРИАНТ №9

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства spawn*(), запускает процесс P2(M2), передавая ему в качестве параметра chid созданного канала, затем переходит в состояние приема сообщений по созданному каналу.

Процесс P2 создает свой канал, и, используя функцию семейства spawn*(), запускает процесс P3(M3), передавая ему в качестве параметра chid созданного

канала, затем устанавливает соединение с каналом процесса P1, передает ему chid своего канала, затем переходит в состояние приема сообщений по своему каналу. Процесс P3 устанавливает соединение с каналом процесса P2 и посылает ему запрос на получение pid процесса P1 и chid его канала. Получив от P2 ответ (pid и chid), устанавливает соединение с каналом процесса P1 и посылает ему свой pid и chid своего канала, после чего переходит в состояние приёма сообщений по своему каналу.

Процесс P1 устанавливает соединение с каналом процесса P2 и передает ему сообщение "P1 send message to P2", получив ответ, выводит его на терминал, затем устанавливает соединение с каналом процесса P3 и передает ему сообщение "P1 send message to P3", получает ответ, выводит его на экран и терминируется.

Процесс P2, получив сообщение от процесса P1, выводит его на терминал, посылает ответ "P2 executed " и терминируется. Процесс P3, получив сообщение от процесса P1, выводит его на экран, посылает ответ "P3 executed " и терминируется.

ВАРИАНТ №10

Разработать приложение, состоящее из четырёх взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства spawn*(), запускает процессы P2(M2), P3(M2) и P4(M3), передавая им в качестве параметра chid1 своего созданного канала, затем переходит в ожидание сообщений по своему каналу от процессов P2, P3 и P4 для обслуживания их запросов.

Процессы P2 и P3 создают свои каналы, устанавливают соединения с каналом процесса P1 - chid1, передают ему дескрипторы своих созданных каналов - chid?, и переходят в состояние приёма сообщений по своим каналам.

Процесс P4 устанавливает соединение с каналом процесса P1, посылает ему запрос на получение дескрипторов процессов P2 и P3, а также дескрипторов ими созданных каналов и ожидает ответа. Процесс P4, получив от P1 ответ с запрошенной информацией о процессах P2 и P3, устанавливает соединение с каналами процессов P2 и P3, посылает им сообщение "P4 send message to P?" (? – номер соответствующего процесса), ждёт и получает ответ от соответствующего процесса P?. После такого взаимодействия с обоими процессами P2 и P3 процесс P4 выводит на экран "P4 executed" и терминируется.

Процессы P2 и P3 после получения сообщения от P4 выводят его на экран и отправляют ответ, затем посылают процессу P1 сообщение "P? received message from P4", после получения ответа выводит на экран "P? executed" и терминируется.

Процесс P1, приняв сообщения от процессов P2 и P3, выводит их на экран и отправляет ответы. После взаимодействия с процессами P2 и P3, процесс P1 выводит на экран "P1 executed" и терминируется.

Приложение

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА (САМАРСКИЙ УНИВЕРСИТЕТ)»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ
Кафедра программных систем

Дисциплина
Технологии промышленного программирования

ОТЧЁТ

по лабораторной работе

Запуск и организация взаимодействия параллельных процессов

Вариант №___

Студент: Фамилия И.О.

Группа: Номер

Преподаватель: Фамилия И.О.

Самара 20___