**Комплект оценочных материалов по дисциплине
 «Операционные системы реального времени»**

**Задания закрытого типа**

**Задания закрытого типа на выбор правильного ответа**

*Выберите один правильный ответ*

1. Какая из перечисленных функций не относится к стандартам POSIX и является специфическим расширением QNX:

А) pthread\_create()

Б) pthread\_sleepon\_lock()

В) pthread\_mutex\_lock()

Г) pthread\_cond\_wait()

Д) pthread\_barrier\_wait()

Правильный ответ: Б

Компетенции (индикаторы): ПК-1.1, ОПК-2.1

2. Какой из переходов в графе состояния процесса практически не возможен:

А) готовность 🡪 выполнение

Б) выполнение 🡪 готовность

В) готовность 🡪 ожидание

Г) ожидание 🡪 готовность

Правильный ответ: В

Компетенции (индикаторы): ПК-1.1, ОПК-11.1

*Выберите все правильные варианты ответы*

3. Укажите все функции QNX предназначенные для реализации роли сервера:

А) ConnectAttach()

Б) ChannelCreate()

В) MsgSend()

Г) MsgReceive()

Д) MsgReply()

Правильные ответы: Б, Г, Д

Компетенции (индикаторы): ПК-3.2, ОПК-2.1

4. Укажите свойства вычислительной системы, существенные для реализации на ее основе системы реального времени:

А) как можно большая вычислительная мощность процессора

Б) наличие как можно большего количества уровней прерываний

В) наличие механизмов виртуальной памяти и страничного обмена

Г) как можно меньшее время реакции на прерывание

Правильные ответы: Б, Г

Компетенции (индикаторы): ОПК-11.1

**Задания закрытого типа на установление соответствия**

*Установите правильное соответствие.*

*Каждому элементу левого столбца соответствует только один элемент правого столбца.*

1. Установите соответствие между предложенными стандартами и регулируемыми ими вопросами:

|  |  |
| --- | --- |
| 1) POSIX.1 (IEEE 1003.1) | А) Расширения для поддержки реального времени |
| 2) POSIX.4 (IEEE 1003.1b) | Б) Независимый от протокола интерфейс сокетов |
| 3) POSIX.4a (IEEE 1003.1c) | В) Базовый API операционных систем |
| 4) POSIX.12 (IEEE 1003.1g) | Г) Потоки внутри POSIX-процессов |

Правильный ответ: 1-В, 2-А, 3-Г, 4-Б

Компетенции (индикаторы): ПК-1.1

2. Установите соответствие между компонентами системного программного обеспечения QNX и их назначением

|  |  |
| --- | --- |
| 1) mkifs | А) запись стадий IPL1 и IPL2 на диск |
| 2) diskboot | Б) автозапуск photon и/или консольных интерактивных процессов |
| 3) dload | В) построение загрузочного образа целевой системы |
| 4) tinit | Г) инициализация раздела QNX и запись загрузочного образа |
| 5) dinit | Д) автомонтирование и загрузка дисковой системы |

Правильный ответ: 1-В, 2-Д, 3-А, 4-Б, 5-Г

Компетенции (индикаторы): ПК-1.1

3. Установите соответствие между заголовочными файлами и предоставляемыми ими интерфейсами:

|  |  |
| --- | --- |
| 1) stdio.h | А) POSIX совместимые операции, такие как read(), fork() и sleep() |
| 2) unistd.h | Б) родные интерфейсы QNX, такие как TimerTimout() и MsgReply() |
| 3) pthread.h | В) независимые от ОС операции ввода-вывода языка C |
| 4) sys/neutrino.h | Г) переносимый интерфейс потоков внутри процессов |
| 5) sys/wait.h | Д) синхронизация процессов POSIX , такая как waitpid() |

Правильный ответ: 1-В, 2-А, 3-Д, 4-Б, 5-Г

Компетенции (индикаторы): ПК-3.2

4. Установите соответствие между состояниями процесса и их характеристиками:

|  |  |
| --- | --- |
| 1) «остановлен» | А) дальнейшее выполнение процесса не возможно, но он еще не удален операционной системой, так как на него имеются ссылки из других процессов |
| 2) «терминирован» | Б) процесс не может получить доступ к процессору, так как в данный момент выполняется более приоритетный процесс |
| 3) «заблокирован» | В) процесс не использует процессор, в таком состоянии процесс находится сразу после создания |
| 4) «готов» | Г) процесс использует процессор |
| 5) «выполняется» | Д) процесс ждет некоторого события, которым может быть аппаратное или программное прерывание, сигнал или другая форма межпроцессорного взаимодействия |

Правильный ответ: 1-В, 2-А, 3-Д, 4-Б, 5-Г

Компетенции (индикаторы): ОПК-11.1

**Задания закрытого типа на установление правильной последовательности**

*Установите правильную последовательность. Запишите правильную последовательность букв слева направо.*

1. Установите правильную последовательность действия при ожидании данных на условной переменной：

А) пока нет данных pthread\_cond\_wait()

Б) pthread\_mutex\_lock()

В) работа с частными данными

Г) работа с общими данными

Д) pthread\_mutex\_unlock()

Правильный ответ: Б, А, Г, Д, В

Компетенции (индикаторы): ПК-3.2, ОПК-11.1

2. Установите порядок событий при загрузке QNX 6.x на x86 системе со стандартным BIOS:

А) стадия IPL2

Б) photon

В) /.boot образ

Г) стадия IPL1

Д) /etc/rc.d/rc.sysinit

Е) diskroot

Ж) tinit

Правильный ответ: Г, А, В, Е, Д, Ж

Компетенции (индикаторы): ОПК-11.1

3. Укажите последовательность событий, приводящих к проблеме инверсии приоритетов:

А) управление получает низкоприоритетная нить, которая освобо­ждает ресурс

Б) нить с наивысшим приоритетом ожидает освобождения ресурса, занятого низкоприоритетной нитью

В) нить с высоким приоритетом может про­должить свою работу, будучи задержанной нитью с промежуточным приоритетом

Г) нить с промежу­точным приоритетом вытесняет низкоприоритетную нить и работает, пока не завершится

Правильный ответ: Б, Г, А, В

Компетенции (индикаторы): ПК-1.1

4. Расположите в корректном порядке события информационного тракта СРВ и устройства связи с объектом:

А) Сигнал подвергается процедуре аналого-цифрового преобразования.

Б) В подсистеме цифровой обработки выполняется преобразование информации с использованием ресурсов компьютера и специализированных процессоров цифровой обработки

В) На вход системы поступает в общем случае аналоговый сигнал, являющийся реализация случайного процесса.

Г) В блоке подготовки сигнал подвергается предварительной аналоговой обработке.

Д) Выполняется восстановление аналогового сообщения по цифровым отсчетам с допустимой погрешностью.

Е) Последовательность отсчетов от различных измерительных каналов объединяется в общий поток для последующего ввода в компьютер или передачи по каналу связи.

Правильный ответ: В, Г, А, Е, Б, Д

Компетенции (индикаторы): ОПК-2.1

**Задания открытого типа**

**Задания открытого типа на дополнение**

*Напишите пропущенное слово (словосочетание).*

1. Механизм наследования приоритета позволяет решить проблему \_\_\_\_\_\_\_\_\_, возникающую при совместном доступе к ресурсу процессов (потоков) с разным приоритетом.

Правильный ответ: инверсии приоритетов

Компетенции (индикаторы): ОПК-11.1, ПК-2.3

2. Те места в программах, в которых происходит обращение к ресурсам, которые не допускают одновременного использования несколькими процессами, называются \_\_\_\_\_\_\_\_.

Правильный ответ: критическими секциями

Компетенции (индикаторы): ОПК-11.1, ПК-2.3

3. Вид многозадачности, при котором высокоприоритетная задача, как только для нее появляется работа, немедленно прерывает низкоприоритетную, называется \_\_\_\_\_\_\_\_\_\_.

Правильный ответ: вытесняющей

Компетенции (индикаторы): ОПК-11.1, ПК-2.3

4. Принудительная передача управления от выполняемой программы к системе (а через нее - к соответствующей программе обработки), происходящая при возникновении определенного события, называется \_\_\_\_\_\_\_\_\_.

Правильный ответ: прерывание

Компетенции (индикаторы): ОПК-11.1, ПК-2.3

**Задания открытого типа с кратким свободным ответом**

*Напишите пропущенное слово (словосочетание).*

1. Технология, при которой, когда в результирующем загрузочном модуле проставляются лишь ссылки на код необходимых библиотечных функций, а сам код будет реально добавлен к загрузочному модулю только при его исполнении, называются \_\_\_\_\_\_\_\_\_\_\_\_\_.

Правильный ответ: динамическое связывание / позднее связывание

Компетенции (индикаторы): ОПК-11.1

*Дайте ответ на вопрос.*

2. При запуске программы с именем hope из текущего каталога в QNX 6.x вы получили ошибку “Segmentation fault. Core dumped”. Программа скомпилирована с опцией –g и исходный код программы также находится в текущем каталоге. Какая команда позволит определить место ошибки.

Правильный ответ: gdb hope /var/dump/hope.core

Компетенции (индикаторы): ПК-2.3

3. После установки QNX 6.x использует однопроцессорную версию ядра. В систему добавлен второй процессор. С помощью какой команды можно обеспечить запуск по умолчанию SMP версии ядра при следующей загрузке системы.

Правильный ответ: cp /boot/fs/qnxbasesmp.ifs /.boot

Компетенции (индикаторы): ОПК-2.1

4. Какой командой QNX можно подключить дискету с файловой системой FAT-12 как каталог /fs/floppy.

Правильный ответ: mount -t dos /dev/fd0 /fs/floppy

Компетенции (индикаторы): ОПК-2.1

**Задания открытого типа с развернутым ответом**

1. Напишите программу, демонстрирующую таймауты QNX на примере блокирующих операций ввода-вывода POSIX, которая позволяет пользователю ввести целое число в течение 5 секунд, и печатает его, либо сообщает об ошибке при ввода, либо о том, что время вышло. Почему нужно использовать связку read() плюс sscanf(), а не просто scanf()?

Время выполнения – 60 мин.

Ожидаемый результат:

Таймауты QNX автоматически деактивируются после каждого вызова ядра, а так как библиотечная реализация scanf() не гарантирует, что ее первым обращением к ядру будет именно read(), то таймаут может быть использован впустую.

#include <stdio.h>

#include <memory.h>

#include <errno.h>

#include <sys/neutrino.h>

#include <unistd.h>

int main() {

 printf("You have 5 seconds to enter a value.\n");

 printf("Enter decimal number: ");

 fflush(stdout);

 fflush(stdin);

 int d;

 char buf[64];

 memset(buf, sizeof(buf), 0);

const unsigned long long SPAN\_5\_SEC = 5 \* 1000000000LLU;

 TimerTimeout(CLOCK\_REALTIME,

\_NTO\_TIMEOUT\_SEND | \_NTO\_TIMEOUT\_REPLY,

NULL, &SPAN\_5\_SEC, NULL

);

 if (read(fileno(stdin), buf, sizeof(buf)-1) > 0) {

 if (sscanf(buf, "%d", &d) == 1) {

 printf("You number is %d\n", d);

 } else {

 printf("Invalid input\n");

 }

 } else {

 printf("\nYou failed to enter number in time\n");

 }

 return 0;

}

Критерии оценивания:

- включение необходимых заголовочных файлов

- корректное задание 5 секундного интервала

- правильное задание причины таймаута при обращении к серверу

- правильное совместное использование функций языка С и вызовов POSIX

Компетенции (индикаторы): ПК-3.2, ОПК-14.3

2. Напишите программу, демонстрирующую межпроцессное взаимодействие с использованием механизмов сообщений QNX в рамках одной ноды. Основой процесс должен порождать субпроцесс и передавать ему данные по единственному запросу со стороны субпроцесса. Основной процесс должен дождаться завершения субпроцесса. Как вы организуете передачу идентификатора родительского процесса и канала связи в дочерний процесс?

Время выполнения – 65 мин.

Ожидаемый результат:

Так как собственный идентификатор известен родителю сразу, а канал связи он открывает до запуска дочернего процесса, то для передачи этой информации проще всего использовать механизм копирования адресного пространства при вызове fork().

#include <stdio.h>

#include <unistd.h>

#include <sys/neutrino.h>

#include <sys/wait.h>

#include <errno.h>

void secondary(int server\_pid, int chid) {

 int coid = ConnectAttach(0, server\_pid, chid, 0, 0);

 if (coid != -1) {

 printf("Secondary: coid = %d, let's send a message\n", coid);

 int dataout = 42, datain = 0;

 int ret = MsgSend(coid, &dataout, sizeof(dataout),

&datain, sizeof(datain));

 if (ret == -1) {

 printf("Secondary: datain = %d\n", datain);

 }

 ConnectDetach(coid);

 }

}

int main() {

 int mypid = getpid();

 int chid = ChannelCreate(0);

 if (chid == -1)

 exit(1);

 printf("Main: pid = %d, chid = %d\n", mypid, chid);

 int pid = fork();

 if (pid == 0) {

 secondary(mypid, chid);

 \_exit(0);

 } else if (pid == -1) {

 exit(1);

 }

 printf("Main: secondary running as pid = %d\n", pid);

 int rcvid, data;

 while ((rcvid = MsgReceive(chid, &data, sizeof(data), NULL)) >= 0) {

 if (rcvid == 0) {

 printf("Main: Pulse ignored");

 } else {

 printf("Main: rcvid = %d, data = %d\n", rcvid, data);

 ++data;

 int ret = MsgReply(rcvid, EOK, &data, sizeof(data));

 if (ret != -1) {

 printf("Main: reply Ok\n");

 }

 break; // только одно сообщение

 }

 }

 printf("Waiting for secondary\n");

 waitpid(pid, NULL, WEXITED);

 ChannelDestroy(chid);

 return 0;

}

Критерии оценивания:

- включение необходимых заголовочных файлов

- корректное порождение и ожидание завершения дочернего процесса

- канал время жизни канала по отношению к дочернему процессу

- правильное ожидание сообщение с игнорированием пульсов

- наличие ответа от сервера

Компетенции (индикаторы): ОПК-14.3, ПК-3.2

3. Напишите программу, моделирующую решение проблемы «производитель-потребитель» с применением POSIX механизма условных переменных. Производитель работает в основном потоке, подготовка данных занимает 700 мсек., общее число блоков данных – 2. Обработка данных занимает 1300 мсек., по этому используется четырехпроцессорная система с тремя потребителями в дополнительных POSIX потоках. Порядок обработки блоков данных не существенен. Производитель должен выводить предупредительное сообщение в случае перегрузки системы. Убедитесь что последний блок данных будет обработан.

Время выполнения – 65 мин.

Ожидаемый результат:

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

const int producer\_delay = 700;

const int consumer\_delay = 1300;

pthread\_cond\_t cond = PTHREAD\_COND\_INITIALIZER;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

const int END\_OF\_DATA = -1;

const int NO\_DATA\_YET = 0;

volatile int the\_data = NO\_DATA\_YET;

void \*consumer(void \*arg) {

 char \*name = (char\*)arg;

 printf("Consumer %s started \n", name);

 for (;;) {

 pthread\_mutex\_lock(&mutex);

 while (the\_data == NO\_DATA\_YET) {

 printf("Consumer %s waiting for new data\n", name);

 pthread\_cond\_wait(&cond, &mutex);

 }

 int local\_data = the\_data;

 if (local\_data > 0) {

 printf("Consumer %s accept data\n", name);

 the\_data = NO\_DATA\_YET;

 }

 pthread\_mutex\_unlock(&mutex);

 if (local\_data == END\_OF\_DATA)

 break;

 printf("Consumer %s processing data block %d\n", name, local\_data);

 delay(consumer\_delay);

 }

 printf("Consumer %s done, name);

}

int main() {

 pthread\_t tid1, tid2, tid3;

 pthread\_create(&tid1, NULL, consumer, (void\*)"A");

 pthread\_create(&tid2, NULL, consumer, (void\*)"B");

 pthread\_create(&tid3, NULL, consumer, (void\*)"C");

 for (int i = 1; i <= 10; ++i) {

 printf("Producer acquiring data block %d\n", i);

 delay(producer\_delay);

 printf("New data ready, posting\n");

 pthread\_mutex\_lock(&mutex);

 if (the\_data != NO\_DATA\_YET)

 printf("WARNING: system overloaded!\n");

 the\_data = i;

 pthread\_cond\_signal(&cond);

 pthread\_mutex\_unlock(&mutex);

 }

 printf("Let consumer peak up last block\n");

 while (the\_data != NO\_DATA\_YET)

 delay(producer\_delay);

 printf("Broadcasting STOP request\n");

 pthread\_mutex\_lock(&mutex);

 the\_data = END\_OF\_DATA;

 pthread\_cond\_broadcast(&cond);

 pthread\_mutex\_unlock(&mutex);

 printf("Main: waiting for consumers to finish\n");

 pthread\_join(tid1, NULL);

 pthread\_join(tid2, NULL);

 pthread\_join(tid3, NULL);

 return 0;

}

Критерии оценивания:

- включение заголовочных файлов для потоков и синхронизации POSIX

- корректная инициализация мьютекса и условной переменной

- умение запустить поток и дождаться завершения

- корректное ожидание данных

- корректная сигнализация готовности, в т.ч. использование broadcast

- применение механизм pthread для финальной синхронизации

Компетенции (индикаторы): ПК-3.2, ОПК-14.3

4. Напишите программу, моделирующую синхронизацию двух дополнительных потоков с основным, с применение механизмов барьеров. Работа потоков должна состоять из трех последовательных стадий с синхронизацией на границе. Выполняемую потоками работу имитировать вызовом delay() c задержками 700 и 1200 мсек. Основной поток выводит информацию о текущем прогрессе.

Время выполнения – 60 мин.

Ожидаемый результат:

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

const int STAGE\_COUNT = 3;

int delay1 = 700, delay2 = 1300;

pthread\_barrier\_t bar;

void \*worker(void \*arg) {

 int delay = \*(int\*)arg;

 int tid = pthread\_self();

 printf("Worker %d started\n", tid);

 for (int i = 1; i <= STAGE\_COUNT; ++i) {

 ::delay(delay);

 printf("Worker %d finished stage %d\n", tid, i);

 pthread\_barrier\_wait(&bar);

 }

 printf("Worker %d is done\n", tid);

}

int main() {

 pthread\_barrier\_init(&bar, NULL, 3);

 pthread\_t id1, id2;

 pthread\_create(&id1, NULL, worker, &delay1);

 pthread\_create(&id2, NULL, worker, &delay2);

 for (int i = 1; i <= STAGE\_COUNT; ++i) {

 printf("Main thread waiting for stage %d\n", i);

 pthread\_barrier\_wait(&bar);

 }

 pthread\_join(id1, NULL);

 pthread\_join(id2, NULL);

 pthread\_barrier\_destroy(&bar);

 return 0;

}

Критерии оценивания:

- включение заголовочных файлов для потоков с барьерами и задержки

- инициализация и разрушение барьерной переменной

- умение запустить поток и передать в него данные

- использование механизмов барьеров для промежуточной синхронизации

- применение механизм pthread для финальной синхронизации

Компетенции (индикаторы): ПК-3.2, ОПК-14.3