1. **Что такое поток управления OS?**

**Поток (управления) OS** – объект ядра операционной системы, которому OS выделяет процессорное время. Наименьшая единица работы ядра OS. Поток (управления) OS – последовательность инструкций, выполняемых процессором в выделенные OS интервалы времени.

Поток создается с помощью системного вызова. При создании процесса, автоматически создается основной (main) поток (выполняется функция ядра, создающая поток).

1. **С помощью каких системных вызовов создаются потоки в Windows и Linux?**

CreateThread в Windows

pthread\_create в Linux

1. **Что такое системные и пользовательские потоки?**

**Поток пользовательского уровня (user thread)**- высокоуровневый *поток*, *операции* над которым включены в *интерфейс* пользователя ОС.

**Поток ядра (kernel thread)**- низкоуровневый системный *поток*, поддержанный и использующийся на уровне ядра операционной системы; используется для реализации **потоков пользовательского уровня**.

1. **Что такое многопоточность?**

**Многопоточность** – модель (парадигма) программирования (OS, системы программирования, программы) поддерживать потоки управления.

**Многопото́чность** — свойство платформы или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени.

1. **Чем отличаются приоритетная многопоточность от и кооперативной многопоточности?**

Операционные системы планируют выполнение потоков одним из двух способов:

1. **Приоритетная многопоточность** позволяет операционной системе определить, когда должно происходить переключение контекста. Недостаток приоритетной многопоточности состоит в том, что система может сделать переключение контекста в неподходящее время, что приводит к инверсии приоритета и другим негативным эффектам, которых можно избежать, применяя кооперативную многопоточность.
2. **Кооперативная многопоточность** полагается на сами потоки и отказывается от управления, если потоки выполнения находятся в точках остановки. Это может создать проблемы, если поток выполнения ожидает ресурс, пока он не станет доступным. Тип многопоточности, при котором следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам.
3. **Что такое диспетчеризация потоков управления OS?**

*Диспетчеризация* – алгоритм, устанавливающий порядок обслуживания очередей потоков процессорами.



1. **Что такое контекст потока и для чего он нужен?**

Контекст потока – данные, необходимые для возобновления работы потока при его приостановке. Содержит программный код, набор регистров, стек памяти, стек ядра ОС, маркер доступа.

1. **Перечислите состояния, в которых может быть поток и поясните их назначение.**

Модель 12 состояний потока:

**New** (поток находится в процессе создания)­-(create)->**Ready** (поток создан, но не выполняется)-:

1. -(run)-> **Running** (выполнение потока)-:
   1. -(exit)-> **Finished** (поток завершён);
   2. -(sleep)-> **Sleeping** (поток приостановлен на промежуток времени)-:
      1. –(suspend)-> **Suspend** **Sleeping** (поток приостановлен на заданный промежуток времени, ожидание завершения промежутка времени приостановлено)-(resume)-> **Sleeping**;
      2. –(wake up)->**Ready**;
   3. –(suspend)-> **Suspend** **Ready** (поток приостановлен)-(resume)-> **Ready**;
   4. –(block)->**Blocked**(поток остановлен до совершения какого-либо события)-:
      1. –(suspend)-> **Suspend** **Blocked** (поток приостановлен до совершения события, дополнительно остановлен)-(resume)-> **Blocked**;
      2. –(unblock)->Ready.
2. -(*interrupt*)->**Ready**(прерывание по окончанию кванта);



1. **Что такое LWP?**

LWP - light-weight process

Легковесным процессом является процесс, поддерживающий работу потока пространства пользователя. Каждый поток пространства пользователя неразрывно связан с легковесным процессом. У пользовательского процесса может существовать ряд связанных легковесных процессов с одинаковым идентификатором группы (group ID). Группировка позволяет ядру производить разделение ресурсов (ресурсы включают в себя адресное пространство, страницы физической памяти (VM), обработчики сигналов и дескрипторы файлов).

1. **Что такое потокобезопасность программного кода?**

**Потокобезопасность кода (программы) –** свойство программного кода (программы) корректно работать в нескольких потоках одновременно.

1. **Что такое реентерабельность кода?**

**Реентерабельность кода (программы)** – свойство одной копии программного кода работать в нескольких потоках одновременно. Реентерабельный код всегда потокобезопасен. Реентерабельный код не использует статическую память и не изменяет сам себя, все данные сохраняются в динамической памяти.

1. **Что такое Fiber?**

Fiber (фибра)– механизм для ручного планирования выполнения кода в рамках потока.

1. **Дайте развернутое определение потока OS.**

**Поток** – средство диспетчеризации доступа к процессорному времени (квант примерно 20мс). Поток представляет из себя последовательность команд процессора, наименьшей единицей работы ядра ОС, которому выделяется процессорное время. Потоки создаются при помощи системного вызова.

Функция **CreateThread** создает поток, который выполняется в пределах виртуального адресного пространства вызывающего процесса.

**HANDLE** **CreateThread**

(

**SECURITY\_ATTRIBUTES** *lpThreadAttributes*,  // дескриптор защиты **SIZE\_T** *dwStackSize*,                        // начальный размер стека **LPTHREAD\_START\_ROUTINE** *lpStartAddress*,    // функция потока **LPVOID** *lpParameter*,                        // параметр потока **DWORD** *dwCreationFlags*,                     // опции создания **LPDWORD** *lpThreadId*                         // идентификатор потока

);

**Параметры**

***lpThreadAttributes***

[in] Указатель на структуру **SECURITY\_ATTRIBUTES**, которая обуславливает, может ли возвращенный дескриптор быть унаследован дочерними процессами. Если ***lpThreadAttributes*** является значением ПУСТО (NULL), дескриптор не может быть унаследован.

***dwStackSize***

[in] Начальный размер стека, в байтах. Система округляет это значение до самой близкой страницы памяти. Если это значение нулевое, новый поток использует по умолчанию размер стека исполняемой программы. Дополнительную информацию см. в статье [**Размер стека потока**](http://narovol.narod.ru/_tbkp/New_MSDN_API/size_stack_thread.htm)

***lpStartAddress***

[in] Указатель на определяемую программой функцию типа **LPTHREAD\_START\_ROUTINE**, код которой исполняется потоком и обозначает начальный адрес потока. Для получения дополнительной информации о функции потока, см. **[ThreadProc](http://narovol.narod.ru/_tbkp/New_MSDN_API/fn_threadproc.htm)**.

***lpParameter***

[in] Указатель на переменную, которая передается в поток.

***dwCreationFlags***

[in] Флажки, которые управляют созданием потока. Если установлен флажок **CREATE\_SUSPENDED**, создается поток в состоянии ожидания и не запускается до тех пор, пока не будет вызвана функция **[ResumeThread](http://narovol.narod.ru/_tbkp/New_MSDN_API/fn_resumethread.htm)**. Если это значение нулевое, поток запускается немедленно после создания. В это время, никакие другие значения не поддерживаются.

***lpThreadId***

[out] Указатель на переменную, которая принимает идентификатор потока.

**Возвращаемые значения**

Если функция завершается успешно, величина возвращаемого значения - дескриптор нового потока.

Если функция завершается с ошибкой, величина возвращаемого значения - ПУСТО (NULL). Чтобы получать дополнительные данные об ошибках, вызовите **[GetLastError](http://narovol.narod.ru/_tbkp/New_MSDN_API/Debbag_error/fn_getlasterror.htm)**[.](http://narovol.narod.ru/_tbkp/New_MSDN_API/Debbag_error/fn_getlasterror.htm)

Функция **TerminateThread** завершает работу потока.

**BOOL** **TerminateThread**(

**HANDLE** *hThread*,  // дескриптор потока

**DWORD** *dwExitCode* // код завершения для потока

);

**Параметры**

***hThread***

[in/out] Дескриптор потока, который завершает работу.

***dwExitCode***

[in] Код завершения работы потока. Используйте функцию **[GetExitCodeThread](http://www.vsokovikov.narod.ru/New_MSDN_API/Process_thread/fn_getexitcodethread.htm)**, чтобы извлечь значение выхода потока.

**Возвращаемые значения**

Если функция завершается успешно, величина возвращаемого значения - не ноль.

Если функция завершается с ошибкой, величина возвращаемого значения - ноль. Чтобы получить дополнительные данные об ошибках, вызовите **[GetLastError](http://www.vsokovikov.narod.ru/New_MSDN_API/Debbag_error/fn_getlasterror.htm)**.

Функция **SuspendThread** приостанавливает работу заданного потока.

**DWORD** **SuspendThread**(

**HANDLE** *hThread* // дескриптор потока

);

**Параметры**

***hThread***

[in] Дескриптор потока, работа которого приостанавливается.

**Возвращаемые значения**

Если функция завершается успешно, величина возвращаемого значения - счет времени предыдущей приостановки работы потока; иначе, оно равно **- (минус) 1**. Чтобы получить дополнительные данные об ошибках, используйте функцию **[GetLastError](http://www.vsokovikov.narod.ru/New_MSDN_API/Debbag_error/fn_getlasterror.htm)**.

Функция **ResumeThread** уменьшает счет времени приостановки работы потока. Когда счет времени приостановки работы уменьшается до нуля, выполнение потока продолжается.

**DWORD** **ResumeThread**(

**HANDLE** *hThread*     // дескриптор потока

);

**Параметры**

***hThread***

[in] Дескриптор для потока, который будет перезагружен.

**Возвращаемые значения**

Если функция завершается успешно, величина возвращаемого значения - предшествующий счет времени приостановки работы потока.

Если функция завершается с ошибкой, величина возвращаемого значения равна **- (минус) 1**. Чтобы получить дополнительные данные об ошибках, вызовите **[GetLastError](http://www.vsokovikov.narod.ru/New_MSDN_API/Debbag_error/fn_getlasterror.htm)**.

**DWORD** — 32-битное беззнаковое целое.

Создание потока происходит с помощью функции pthread\_create(pthread\_t \*tid, const pthread\_attr\_t \*attr, void\*(\*function)(void\*), void\* arg), где: tid - идентификатор потока, attr - параметры потока (NULL - атрибуты по умолчанию, подробности в man), function - указатель на потоковую функцию, в нашем случае threadFunc и arg - указатель на передаваемые данные в поток.

Функция pthread\_join ожидает завершения потока thread. Второй параметр этой функции - результат, возвращаемый потоком.



gcc –D\_REENTRANT –std=c99 xxx.c –o xxx -lpthread