

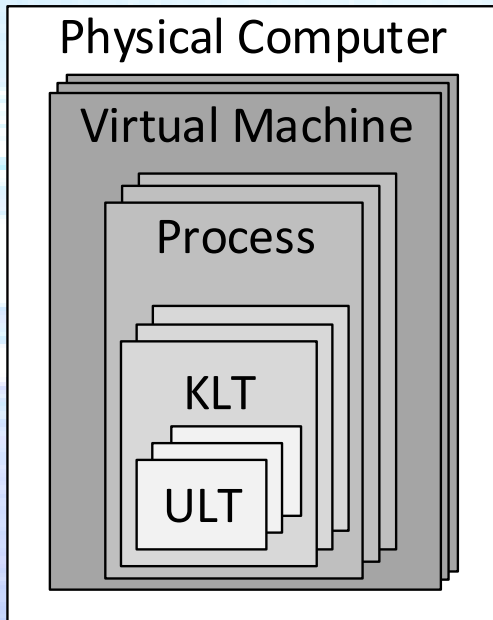
Многозадачность: Процессы и потоки

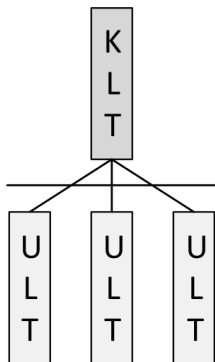
Евгений Иванович Клименков

osisp2019@gmail.com

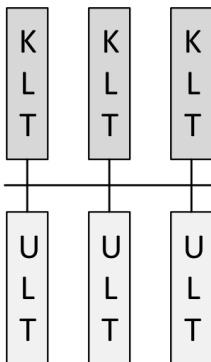
Белорусский Государственный Университет
Информатики и Радиоэлектроники

2019

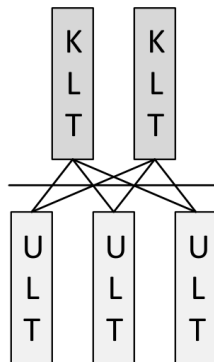




(a) Many-to-one

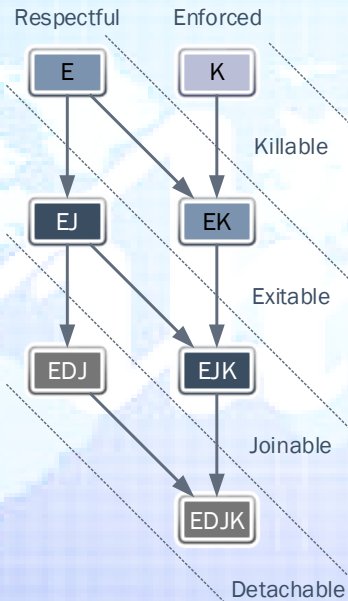


(b) One-to-one



(c) Hybrid

Классификация моделей жизненного цикла

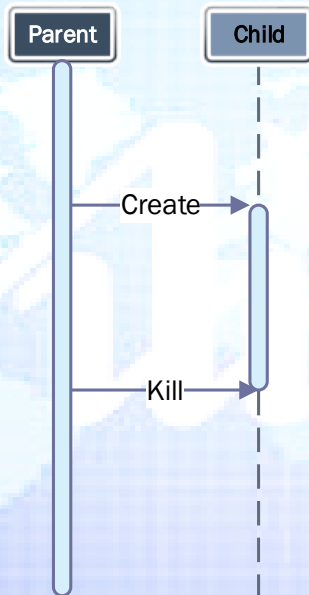


- Создание
- Разрушение
 - Добровольное
 - Принудительное

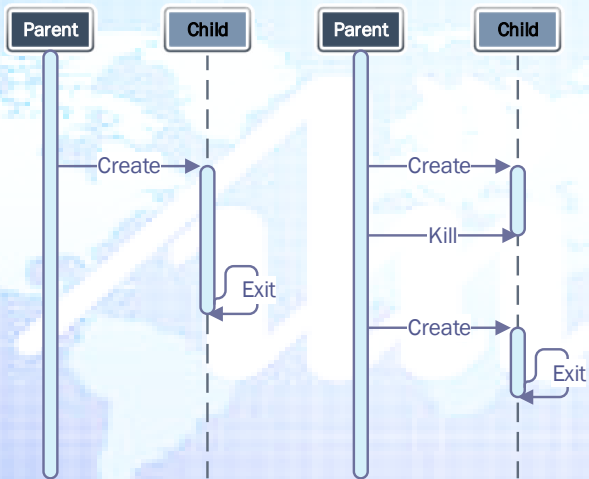
Симметричное API!

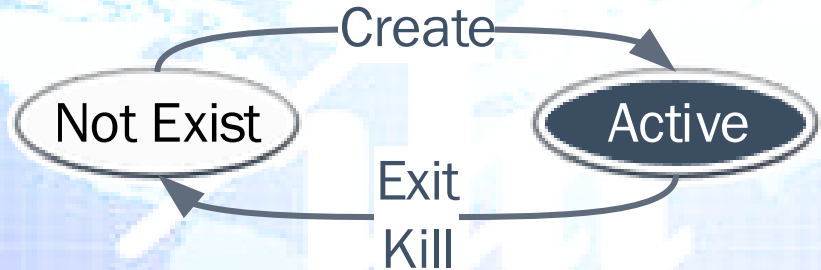
Создание: `CreateTask(entry point, args, scheduling attributes, stack address, stack size, state, source resources)` -> task id

K: Killable Tasks

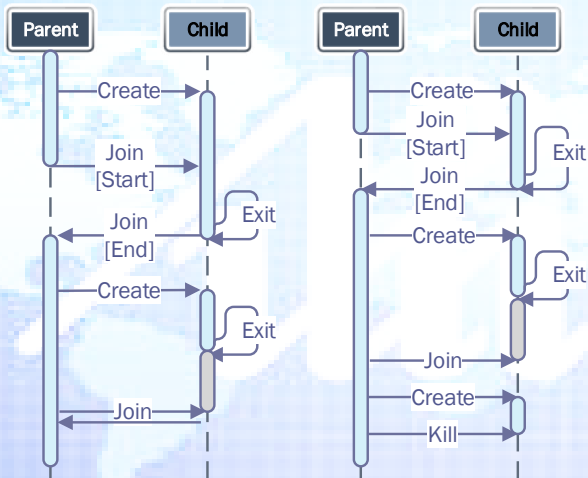


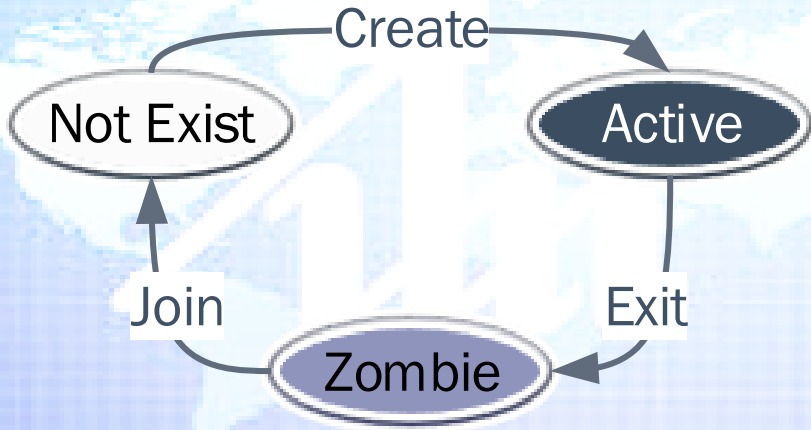
E: Exitable Tasks





E: Joinable Tasks





*NIX Processes: Fork-Exit-Wait

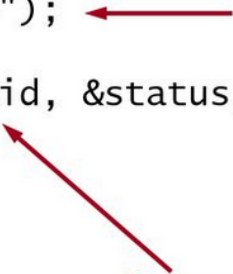
- `pid_t fork(void);`
- `void exit(int status);`
 - `pid_t wait(int *wstatus);`
 - `pid_t waitpid(pid_t pid, int *wstatus, int options);`
- `int execl(const char *path, const char *arg, ... /* (char *) NULL */);`

*NIX Processes: Fork-Exit-Wait

```
int pid = fork();  
if (pid == 0) {  
    exec("foo");  
} else {  
    waitpid(pid, &status, options);  
};
```

Child process

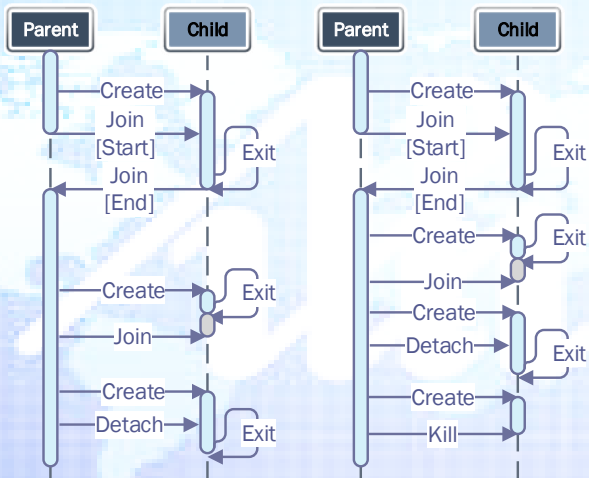
Parent process

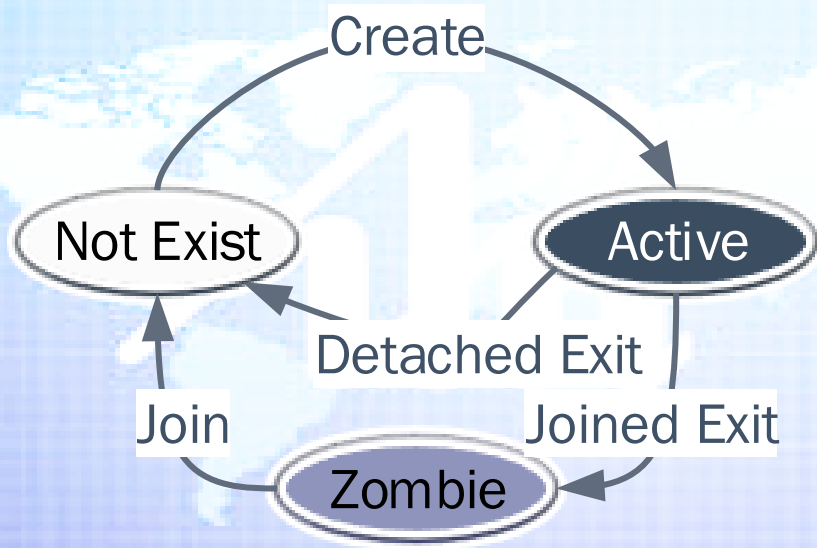


COW: Copy-On-Write

- `int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);`
- `void pthread_exit(void *retval);`
- `int pthread_join(pthread_t thread, void **retval);`

E: Detachable Tasks





- Process table: `ProcessControlBlock processes[]`;
- Thread table: `ThreadControlBlock threads[]`;

TCB:

- Ядерный стек*
- Виртуальное адресное пространство (процесс)
- Контекст процессора
 - FPU регистры
 - SSE регистры
 - CR2 снапшот
 - SP ядерного стека*
- Коммуникационный канал*
 - Адрес
 - Размер
 - Указатель чтения
 - Указатель записи
- Иерархическое положение
 - Родитель
 - Брат-Брат
 - Потомок

Каждый поток в системе имеет:

- Стек пользовательского режима
- Стек ядерного режима
 - Разделяемый
 - Частный (не вытесняемое ядро :-(BKL)

При смене контекста происходит смена активного стека.

ТСВ:

- Опции планирования
 - Дедлайн
 - Квант времени
 - Эпоха
 - Очередь
- Счетчик ссылок
- Обработка исключений
 - Check Point
 - Обработчик исключения
 - Статус обработки
- Статистика

PCB:

- Физический адрес корня таблицы
- Виртуальные адреса элементов таблицы
- Счетчик ссылок
- Права доступа к портам ввода-вывода
- Обработчик(и) отказов доступа к страницам

По шагам:

- 1 Каким-то образом вызывается планировщик
 - принудительное перепланирование
 - добровольное перепланирование
 - полудобровольное перепланирование
- 2 Переход в режим ядра
 - прерывание
 - системный вызов
- 3 Процессор производит переключение
 - проверка прав
 - переключение стека
 - передача управления на код ядра
- 4 Сохранение состояния регистров процессора на стеке
- 5 Выполнение первичного запроса/события
- 6 Инициация события приводящего к перепланированию

Переключение задач

По шагам:

- 1 Вызов планировщика в ответ на это событие
 - что делать с текущим событием?
 - какой поток запустить?
- 2 Переключение контекста (исходный и целевой TCB)
- 3 Определение и сохранение значений активных регистров
- 4 Переключение на новый стек ядра
- 5 Определение и восстановление значений активных регистров
- 6 Переключение адресного пространства (если оно нужно)
- 7 Переключение Thread Local Storage (TLS)
- 8 Установка нового адреса стека для точки входа в режим ядра
- 9 Установка нового значения для системного таймера
- 10 Сбор статистики
- 11 Продолжение прерванного выполнения запроса

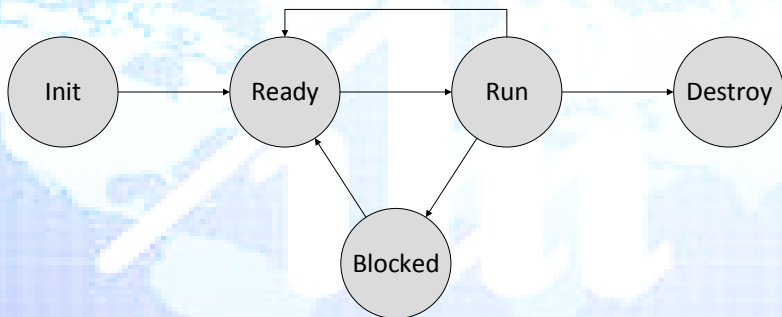
По шагам:

- ① Выход из режима ядра
 - восстановление значений регистров
 - переключение стека
 - передача управления на ранее сохраненную точку пользовательского кода
- ② Продолжение выполнения прерванной программы

Планирование - политика, согласно которой определяется очередность, время выполнения и место выполнения задач.

Планирование - политика, согласно которой определяется очередность, время выполнения и место выполнения задач.

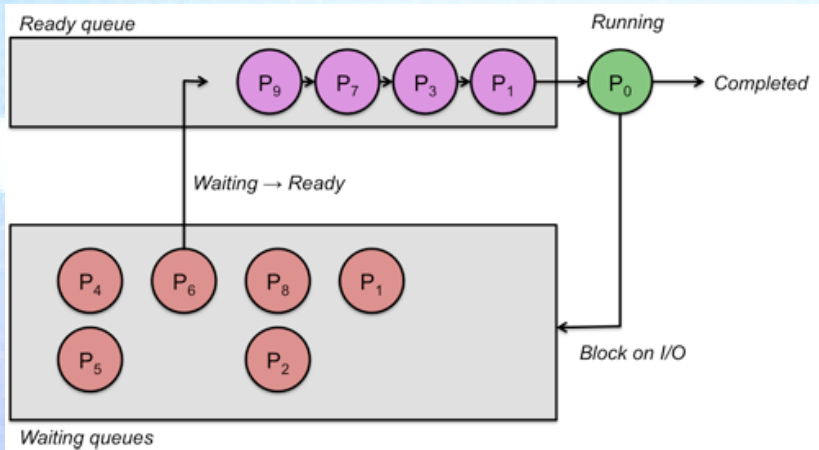
Жизненный цикл задачи



- Статическое планирование
- Динамическое планирование

- Справедливость – каждая задача должна получить справедливую порцию процессорного времени
- Эффективность – процессор не должен простаивать
- Максимизация пропускной способности
- Минимизация времени отклика
- Минимизация накладных расходов
- Максимизация использования ресурсов
- Устранение голодания (starvation)
- Приоритеты
- Равномерная деградация

First-Come, First-Served



- + : Простота, справедливость
- : Интерактивность, Time-to-Completion

Наивысший приоритет имеет самый короткий процесс.

+: Простота, максимальную пропускная способность системы, минимизация среднего времени ожидания

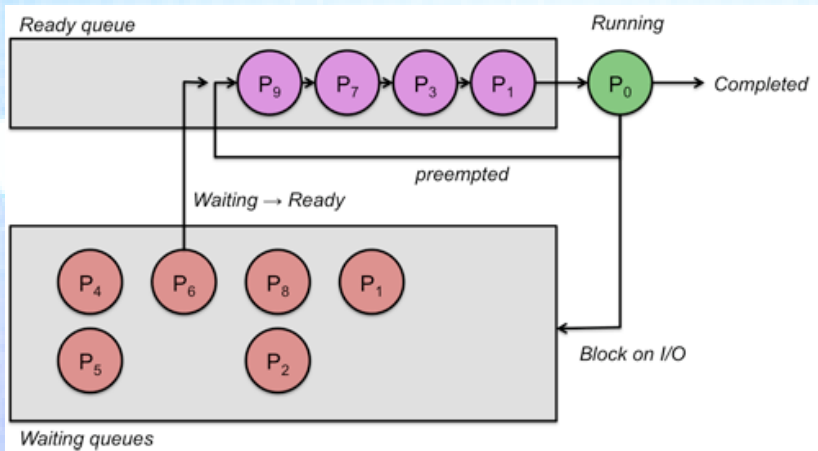
-: Интерактивность, Starvation, Время выполнения задания должно быть известно

Наивысший приоритет имеет процесс с самым коротким оставшимся временем выполнения.

+: Простота, максимальную пропускную способность системы, минимизация времени ожидания

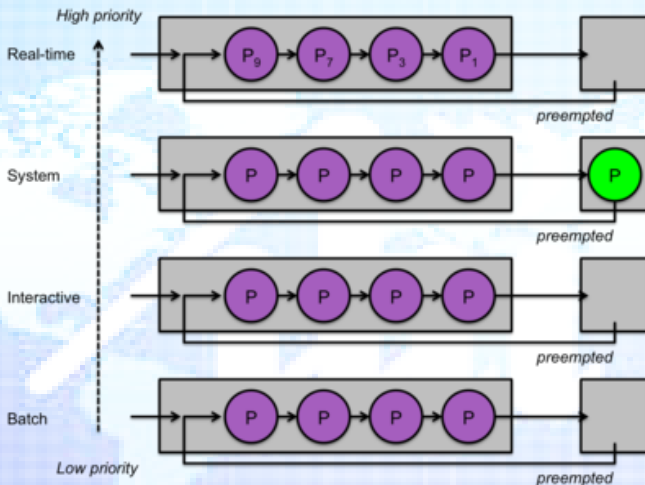
-: Интерактивность, Starvation, Время выполнения задания должно быть известно

Round Robin



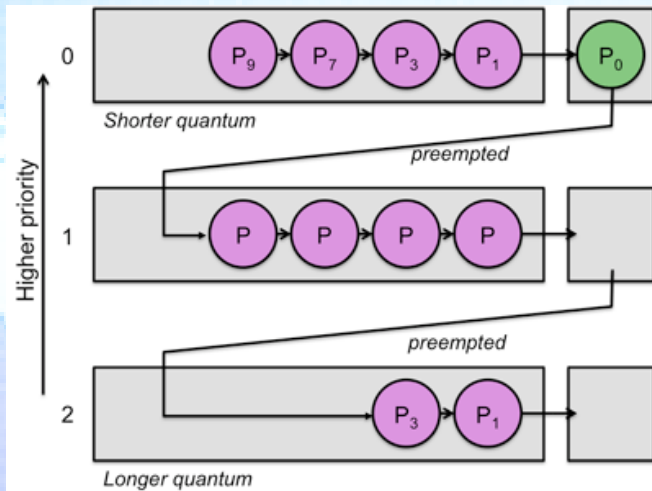
- + : Простота, справедливость, равномерная деградация
- : Интерактивность

Multilevel Queues



- + Простота, справедливость, равномерная деградация, Интерактивность
- : выбор приоритета, Starvation

Multilevel Feedback Queues



- + : Простота, справедливость, равномерная деградация, Интерактивность+
- : выбор приоритета, Starvation-

Lottery Scheduling (Fair Share)

Задачам назначаются “билеты”. Общее количество билетов ограничено и фиксировано. Распределение билетов задает распределение процессорного времени.

+: QoS, Простота

-: Starvation-

Round Robin with Dynamic Quanting

В основе обычный Round Robin.

Две очереди готовых задач (основная и резервная).

Вся история делится на эпохи. В рамках каждой эпохи задачам назначается доля времени в эпохе (динамический квант). При блокировке вызванной исчерпанием кванта, задача заносится в одну очередь готовых задач, при блокировке – в другую. При опустошении активной очереди готовых задач, происходит смена эпохи. Очереди меняются ролями.

+: QoS, Простота

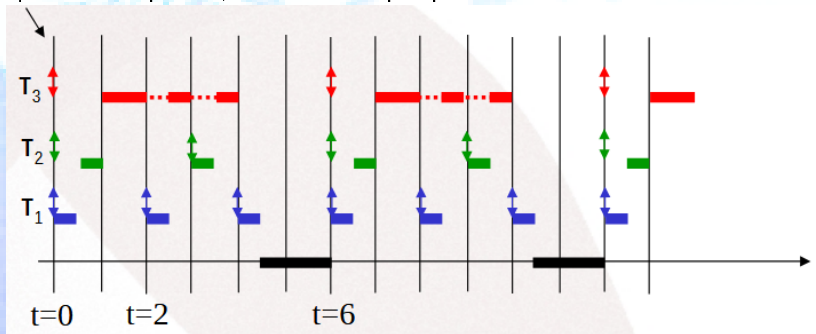
-: Response Time

Отличительные черты ОСРВ

| | ОС реального времени | ОС общего назначения |
|----------------------|--|--|
| Основная задача | Успеть среагировать на события, происходящие на оборудовании | Оптимально распределить ресурсы компьютера между пользователями и задачами |
| На что ориентирована | Обработка внешних событий | Обработка действий пользователя |
| Как позиционируется | Инструмент для создания конкретного аппаратно-программного комплекса реального времени | Воспринимается пользователем как набор приложений, готовых к использованию |
| Кому предназначена | Квалифицированный разработчик | Пользователь средней квалификации |

Rate Monolithic Scheduling

Приоритет определяется цикличностью задачи. Чем короче цикл повторения, тем выше приоритет.



Fixed Priority Scheduling

+ : Масштабируемость + : Динамичность + : Детерменированное поведение при перегрузке
- : Сложность - : Накладные расходы - : Starvation

Следующая задача – задача с ближайшим дедлайном.
Картинки нет :-)

Starvation возникает когда задаче постоянно отказывают в процессорном времени.

Overloading (DoS) – перегрузка системы.

Priority Inversion – Низкоприоритетная задача блокирует высокоприоритетную.