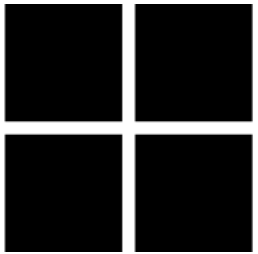
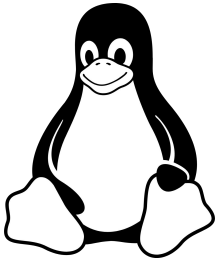
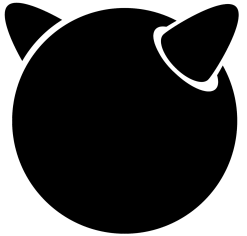
Operating systems





macOS



Процессы и каналы

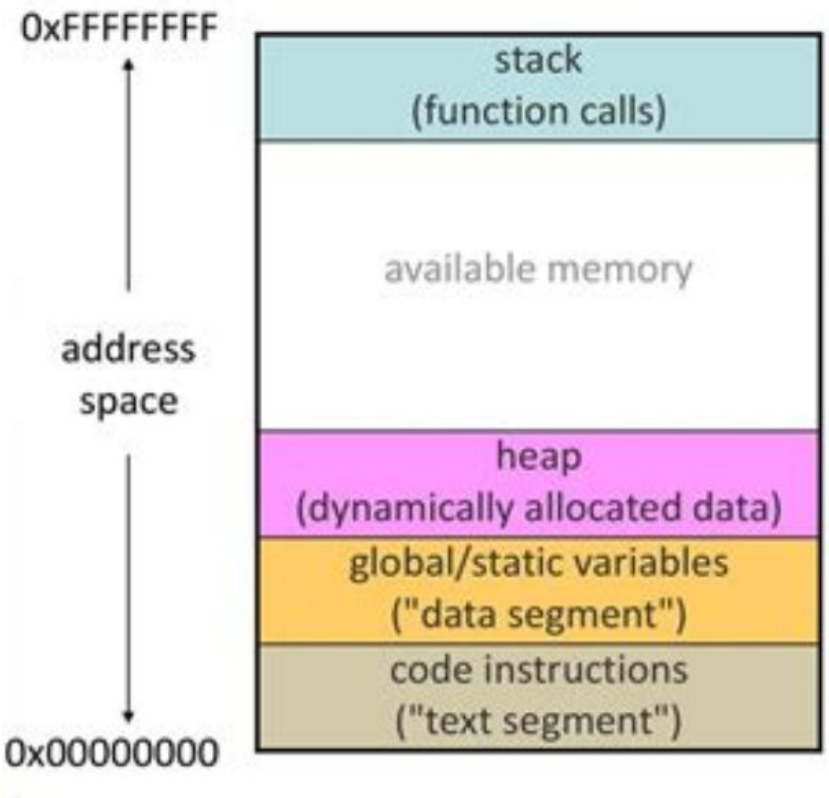
Процесс

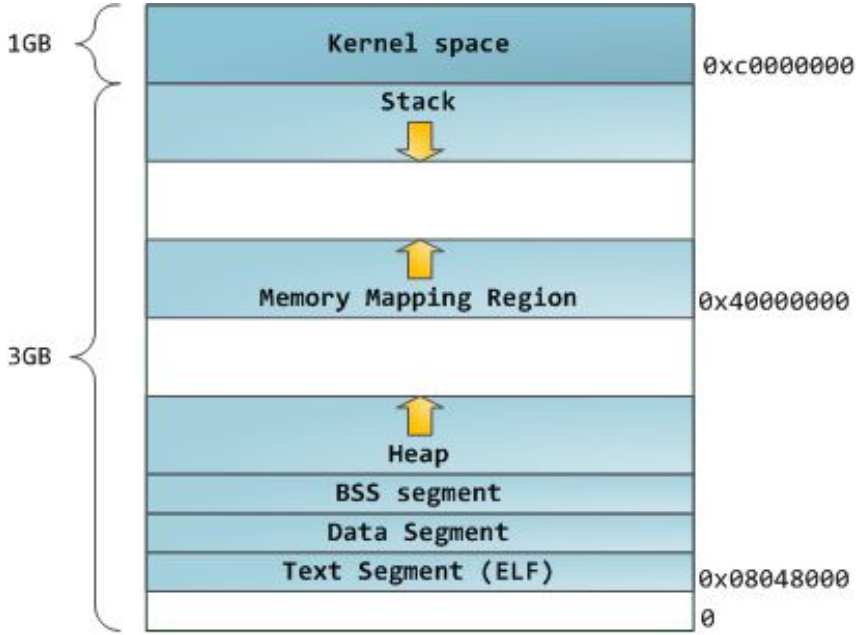
Процесс – это динамический объект внутри вычислительной системы, который представляет собой экземпляр выполняемой программы, имеющей полное состояние регистров процессора (указатель инструкции и регистры общего назначения), а также позволяющей выделить ей ресурсы памяти, устройств и других объектов ОС

Процесс

- Для исполнения одной программы можно организовать несколько процессов
- В рамках одного процесса может выполняться несколько программ (потоки)
- В рамках процесса может исполняться код, отсутствующий в программе, т.е. системные вызовы (syscallы)

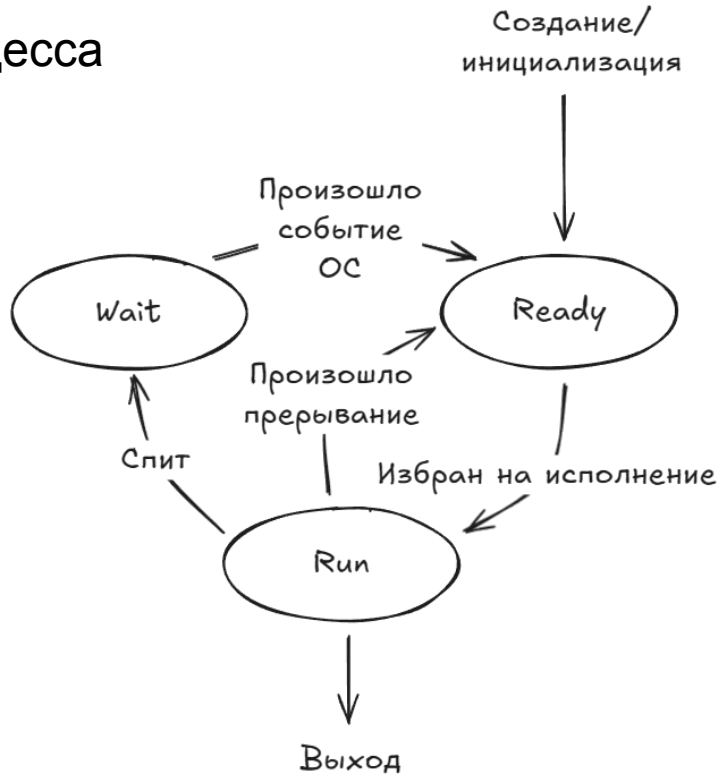
Память процесса





Свойства процесса

1. Состояние процесса



Свойства процесса

2. Операции:

- Создание процесса / выход процесса
- Процесс исполняется / процесс прерван
- Процесс заблокирован / процесс разблокирован

Process Control Block

- Состояние процесса
- Указатель инструкции / программный счетчик
- Содержание регистров
- Данные для планирования использования CPU
- Данные для управления памятью процесса
- Учетная информация (им, пользователь, время, ID и пр.)
- Информация об используемых процессом объектов OS

Регистровый контекст

Системный контекст

- Исполняемый код и данные в адресном пространстве

Контекст процесса

Создание процесса

1. Порождение нового PCB с изначальным состоянием процесса “New”
2. Присвоение ему PID (Process ID)
3. Выделение ресурсов ОС
4. Занесение в адресное пространство кода и установка значения указателя инструкции на начало программы (*main*)
5. Окончание заполнения PCB
6. Выставление состояние в “Ready”

Создание процесса

Занесение в адресное пространство кода и установка значения указателя инструкции на начало программы:

- В Unix системах происходит дубликат родительского процесса (новое адресное пространство, которое шарит только код, и новый PCB); поддерживается иерархия

- В Windows системах происходит прямое создание процесса из исполняемого файла; процессы независимы

Создание процесса

События, инициализирующие процессы:

- Инициализация системы
- Работающий процесс осуществляет системный вызов создания нового процесса
- Создание процесса по запросу пользователя
- Инициализация пакетного задания

Завершение процесса

1. Изменение состояния на “Exit”
2. Освобождение ресурсов
3. Очистка элементов PCB, связанных с исполнением программы
4. Сохранение в PCB информации о причине завершения

Исполнение процесса

1. Выбор одного процесса из набора планировщиком задач OS
2. Перевод состояния в “Run”
3. Восстановить из PCB значения регистров
4. Прыгнуть к инструкции по регистру указателя инструкции и начать/продолжить исполнять программу

Прерывание процесса

1. Сохранить регистры в PCB
2. Прыгнуть по адресу, диктованным таблицей прерываний
3. Сохранить динамические части регистрового и системного контекста PCB
4. Переключиться на стек ядра
5. Обработать прерывание
6. Переключиться на стек процесса
7. Изменить состояние процесса в “Ready”

Блокирование процесса

1. Сохранение системного контекста PCB
2. Обработка системного вызова
3. Перевод состояния процесса в “Wait”

Разблокирование процесса

1. Ядро узнает про событие из очереди внутри OS
2. Проверяет процессы, ожидающие событие
3. Если событие для процессов произошло, то
    - событие обрабатывается (input/output)
    - и состояние процесса становится “Ready”,
    - иначе процессы все еще ждут…

Кооперация процессов

Причины объединения процессов:

- Повышение скорости решения задач (threads vs. processes) в контексте многоядерной системы
- Совместное использование данных
- Модульная конструкция какой либо системы (микросервисы?..)
- Удобство работы пользователя (shell piping)

Процессы влияют на поведение друг друга путем обмена информации

Основные аспекты организации передачи данных

- Нужна ли инициализация?
- Способы адресации: прямая (симм. / асимм.), косвенная
- Пределы средств связи:
  - Сколько процессов может воспользоваться
  - Направленность: simplex, half-duplex, full-duplex
- Особенности каналов:
  - Буферизация: нет, конечной ёмкости, ♾️
  - Модели передачи: потоковая (нет структуры) и модель сообщений (есть структура)

Как работает pipe

- Имеются неименованные (pipe) и именованные (fifo) каналы
- Имеет потоковую модель передачи (структуры нет, обменивайтесь байтами)
- Есть небольшой буфер, который ядро поддерживает у себя и не дает никому к нему random-access доступа
- Общаться могут только два процесса
- В Unix системе каналы являются simplexами, а в Windows – full-duplex
- Адресация каналов является косвенной, по имени или файловому дескриптору

А теперь примерчик!

Нам нужно научиться:

- Создавать процессы
- Создавать каналы
- Открывать файлы
- Обмениваться данными по каналам