Семинар 19. Процессы

Атрибуты процесса

- Идентификатор процесса (pid)
- Идентификатор родительского процесса (ppid)
- Статический (относительный) приоритет процесса (nice number)
- Динамический приоритет процесса

Атрибуты процесса

- Таблицы страниц виртуального адресного пространства
- Разделяемые и неразделяемые страницы памяти
- Отображения файлов в память
- Стек ядра
- Таблица файловых дескрипторов
- Umask процесса

Идентификаторы процессов

- **pid** — идентификатор процесса, положительное целое число [1..32767] (1 — процесс init)
- **ppid** — идентификатор родительского процесса (если родитель процесса завершается, родителем становится init)
- **pgid** — идентификатор группы процессов (группа процессов выполняет одно задание)
- **sid** — идентификатор сессии (сеанса работы)

Идентификаторы процессов

- **getpid** и **getppid**
- Заголовочный файл unistd.h
- Позволяют узнать процессу свои **pid** и **ppid**

Создание процессов

- Для создания процесса используется системный вызов **fork**
- Заголовочный файл - **unistd.h**
- Это единственный способ создания нового процесса
- При ошибке возвращается **-1**, иначе - **pid** созданного процесса
- Внутри нового процесса возвращается 0
- Новый процесс - копия исходного

Создание процессов

- Практически все атрибуты копируются, страницы памяти копируются в режиме copy-on-write
- Не копируются: pid, ppid, сигналы, ожидающие доставки, таймеры, блокировки файлов
- Копируются в том числе и структуры данных, инициализированные стандартной библиотекой

Завершение работы процесса

- **exit** (структуры данных стандартной библиотеки очищаются) и **_exit** (просто системный вызов)
- Получение сигнала, вызывающего завершение (например, **SIGTERM**)
- При получении сигнала также может быть записан образ памяти (core dump)

Завершение работы процесса

- Освобождение страниц памяти, использованных процессом
- Закрытие всех открытых дескрипторов файлов
- Освобождение прочих ресурсов, связанных с процессом, кроме статуса завершения и статистики ресурсов
- Если у процесса есть потомки, родителем потомков назначается процесс init
- Родителю процесса посылается сигнал SIGCHLD

Ожидание завершения процесса

- **pid_t wait(int *status);** - приостановить выполнение до завершения дочернего процесса
- **pid_t waitpid(pid_t pid, int *status, int options);** - приостановить выполнение до завершения дочернего процесса с конкретным pid
- Заголовочный файл **sys/wait.h**

Ожидание процесса

- Процесс-зомби - дочерний процесс в Unix-системе, завершивший свое выполнение, но еще присутствующий в списке процессов операционной системы, чтобы дать родительскому процессу считать код завершения
- Такие процессы не потребляют ресурсов, однако занимают место в таблице процессов

Замещение тела процесса

- Замещение тела процесса — запуск на выполнение другого исполняемого файла в рамках текущего процесса
- Новая программа также наследует от вызвавшего процесса его идентификатор и открытые файловые дескрипторы, на которых не было флага закрыть-при-exec
- Для замещения тела процесса используется семейство exec*: системные вызов **execve** и функции **execv**, **execvp**, **execl**, **execlp**, **execle**
- **v** - передается массив параметров, **l** - передается переменное число параметров, **e** - передается окружение, **p** - выполняется поиск по PATH

Замещение тела процесса

- Сохраняются все атрибуты, за исключением
  - Атрибутов, связанных с адресным пространством процесса
  - Сигналов, ожидающие доставки
  - Таймеров

Системный вызов execve

- **int execve(const char *path, char *const argv[], char *const envp[]);**
- **path** — путь к исполняемому файлу
- **argv** — массив аргументов командной строки, заканчивается элементом **NULL**
- **envp** — массив переменных окружения, заканчивается элементом **NULL**
- Аргументы командной строки и переменные окружения помещаются на стек процесса

Функция execlp

- **int execlp(const char *file, const char *arg, ...);**
- Выполняется поиск исполняемого файла **file** по каталогам, перечисленным в переменной окружения **PATH**
- Аргументы запускаемого процесса передаются в качестве параметров функции execlp
- Последним аргументом функции должен быть **NULL**

Схема fork/exec

- Системный вызов **fork** создает новый процесс
- В дочернем процессе системными вызовами настраиваются параметры процесса (например, текущий рабочий каталог, перенаправления стандартных потоков и пр.)
- Вызовом **exec*** запускается требуемый исполняемый файл

Подготовка аргументов командной строки

- Часто необходимо запустить программу, если передана строка состоящая из имени программы и аргументов
- Для этого можно использовать **int system(const char *command);**
- Другой вариант **execlp("/bin/sh", "/bin/sh", "-c", command, NULL);**