**Лекция 11**

Есть проблема: если у процесса много детей, то непонятно, кого из них он дождется. Если хочется дождаться конкретного ребенка, у wait есть форма:

*pid\_t waitpid(pid\_t pid, int\* wstatus, int options)*

Что будет, если родитель завершится раньше ребенка? Тогда все его дети adopted by the process with pid 1. По факту же, из примера, получилось, что детей приютил процесс systemd, у которого уже ppid = 1.

**Демоны** — процессы в линуксе, которые работают постоянно и мониторят происходящее: следят за запуском и остановкой других процессов, за запуском задач по расписанию и т.п.

**kill pid** — убить процесс с id = pid

Процессы образуют дерево.

**pstree** — команда, с помощью которой можно отобразить иерархию процессов.

**ps** — посмотреть список процессов, запущенных из под терминала

**ps aux** — отобразить список всех процессов

**Зомби** — процессы, которые завершились, но родитель еще их не подобрал (ресурсы процесса почищены, но информация о нем еще хранится в таблице процессов)

Процесс можно не только прервать (ctrl C или ctrl D), но и приостановить (ctrl Z).

Разница между спящим и остановленным процессом в том, что спящий процесс ждет какого-то события, таймера или ввода, а остановленный процесс просто ждет, когда ему скажут продолжить. Когда процесс остановлен, на обычный kill он не реагирует, его нужно убивать с помощью *kill -KILL pid*.

По умолчанию kill без параметров убивает процесс. Но с его помощью можно и посылать другие сигналы, например останавливать процессы: kill -STOP pid.

*Kill -CONT pid* — продолжить процесс

Когда ребенок завершается, родитель получает сигнал SIGCHILD.

**./a.out &** — запустить процесс в фоновом режиме

**jobs** — команда, которая отображает список задач, которые запустили из этого терминала в фоновом режиме

**fg** — команда, которая берет последнюю задачу из фонового режима и выводит ее на передний план

**bg** — перевести остановленный процесс в фоновый режим для выполнения

Всего существует 5 состояний пользовательских процессов в линуксе:

* R — исполняемый или состоящий в очереди на исполнение
* S — спящий, находящийся в ожидании таймера или пользовательского ввода
* T — остановленный
* Z — зомби
* D — состояние непрерываемого сна: процесс ожидает некоторого события, чтобы потом продолжить выполняться, и при этом не реагирует ни на какие сигналы, и его, следовательно, нельзя убить — убьется только перезагрузкой системы

С точки зрения ОС, что такое процесс? Чем программа отличается от процесса?

**Программа** — исполняемый код.

**Процесс** — программа вместе с ее runtime состоянием. Запущенный код с навешанными на него атрибутами, свойствами, некоторой другой информацией. Некоторая структура данных в ядре ОС.

У процессов есть приоритеты, а также свойство **niceness** — число, противоположное приоритету. Приоритет процесса тем выше, чем ниже его приятность. Приятные процессы легко уступают другим. По умолчанию niceness процессов равен 0.

Можно поменять niceness только в сторону уменьшения.

**nice** — команда, которая запускает процесс с определенным niceness. Niсeness меняется в пределах от -20 до 19.

Отрицательный niceness может поставить только root.

*nice -n 11 ./a.out*

**renice** — команда, которая меняет niceness уже запущенного процесса

*renice -n priority pid*

Есть одноименные сисколы nice и renice.

Также у процессов есть атрибуты **uid** и **euid**.

**getuid** — сискол, позволяющий узнать, от имени какого пользователя работает процесс (get real user id)

**geteuid** — get effective user id, то есть по сути сискол дает информацию о том, с привилегиями какого пользователя запущен процесс

*sudo -u user command* — запустить от имени user команду command

**setuid** — сискол, который позволяет процессу начать исполняться из-под другого пользователя

*int setuid(uid\_t uid)*

Если запустить из-под root для какого-нибудь процесса, то права root потеряются и обратно вернуть не получится.

**seteuid** — set effective user or group id

Как вообще получается повысить себе права до root с помощью sudo?

Если посмотреть на права программы sudo, увидим rws (вместо rwx). s — **suid bit**. Некоторым программам в линуксе можно поставить особое право, такое, что всякий раз, когда они запускаются, это выглядит так, будто их запустил их владелец (в случае с sudo это root).

**passwd** — программа, которая позволяет менять пароли, у нее тоже есть suid bit.

Suid bit можно поставить на свою программу: *chmod u+s a.out*

Если сначала убрать права на выполнение, а потом поставить suid bit, то получим на соответствующем месте право S, что означает, что suid bit выставлен, но прав на выполнение у владельца нет, но root все еще сможет запустить эту прогу.

(sudo -s или sudo su — стать root)