ОС. Семинар 6

19.10.2024г.

Демоны

Ф-ция daemonize() делает процесс демоном, вызывается из ф. main() - т.входа.

❗ Прокомментировать строку PPID, PID, PGID, SID, State и т.д. - проработать, уметь объяснить.

В parent'е:

1. umask(0) (без маски) - сброс маски режима создания файлов (права доступа, тип файла и инф. о файле, которая м.б. установлена установочной функцией).
2. fork(0) - создаёт процесс-потомок, который наслед ...[см. сем fork] и маску режима создания файлов. => Потомок унаследует сброшенную маску.

Зачем? { Чтобы потомок мог создать любые файлы без ограничений.

В примере подчёркивается наследование.

3. Завершается процесс-предок для того, чтобы потомок утратил группу.

❗ В Linux: повторный вызов fork() не нужен.

Завершаем предка => потомок теряет груп-

ну, он становится сиротой => его усыновляет терм. процесс => не лидер группы - усл. setsid

setsid - делает демона упр. терминала, делает его лидером группы и лидером сессии, ! и он единственный в группе / сессии. Как в поэме Лермонтова "Демон". Одинокий демон большую часть жизни находится в Інтер. sleep (фоновый процесс).

4. Вызов ф-ции chdir() - т.дир. меняется на корневой каталог, т.к. демон выполняется долг. время

Файл. сист. можно монтировать / отмонтировать (mount). => Надо перейти в корн. каталог, чтобы иметь возможность отмонтировать файловую систему.

5. Закрыть все дескрипторы откр. файлов. Т.к. потомок наслед. дескр. откр. файлов, которые демону не нужны.
   - getrlimit макс. возм. номер дескр.
   - rlim_max = 1024 (которые м.б. открыты)

Как реализован в ядре с.в. open()? В коде библ. ф-ций сист. ф-ции read, write, open.

Системный вызов open() возвращает файловый дескриптор (целое число - int) - дескриптор откр. файла в таблице откр. файлов.

Процессу изначально выделяется таблица размером 512. Для созд. нового файла - сист. должна в таблице найти 1й свободный дескриптор (0,1,2 всегда заняты stdin/out/err). Если в таблице из 512... нет ни одного свободного, то ядро переходит к созд. ещё одной таблицы => => ошиб. Если не хватит - ошибка.

В цикле for закрываются все файловые дескрипторы, начиная с нулевого.

! Файлы stdin/stdout/stderr есть всегда => они открываются на нем устройстве - чёрная дыра, прочитать ничего нельзя.

6) sigaction работает со структурой sig_action, заполняются поля этой структуры хэндлер. значения sigemptyset(mask) - сброс сиг. маски. flags = 0 по умолч.

sigaction (SIGHUP) - инф. разрыв. связи с терм. (чтение конф. файла).

7) Все сообщения записываются в сист. журнал - syslog. LOG_CONS, LOG_DAEMON

8. Демоны д.б. в единственном экземпляре. Т.к. экз. одного демона выполняет одну и ту же работу => могут возникать ошибки, дополн. накладные расходы. LOCKFILE – файл блокировки – нужен, чтобы демон сущ. в ед. экземпляре.

Нужны права суперпользователя, т.к. в дир /var/run для записи нужны права соот.