Семинар. 17.10.2022 // Демон.

Необходимо читать профессионально значимую литературу.

В Unix сигналы – базовое средство информирования процессов о событиях, происходящих в системе. Сигналы включены в классический вариант Unix.

Мониторинг

Важнейшим событием – завершение: нормальное, команда kill, ctrl+c/z и прочее с клавиатуры.

Параллельные системы…

System 5. BSD. Отличаются API, несовместимы.

Процесс – может определить собственную реакцию на получаемый сигнал. Может игнорировать получаемый сигнал, реагировать по умолчанию, определить собственную реакцию на полученный сигнал. В коде 5ой программы необходимо написать собственный обработчик (handler – классическое названия обработчика).

В 5ой программе регистрация нахого обработчика сигнала на указ сигн. Не входит в Posix1. Не рекомендуется использовать в переносимых ПО.

Систем вызов входит в POSIX – sigAction?

Для чего процессу нужно определять свою реакцию на сигнал: собственный обработчик сигнала. С помощью техники сигнала можно менять ход выполнения программы. Именно это у нас в проге и происходит: меняется ход выполнения.

При этом. Системный вызов сигнала входит в анси Си и есть всегда.

Процессы-Демоны.

Книга Стивен Рагуа (13 глава). Чтобы увидеть демонов: #ps -agx.

Parent id0 (процесс init, запустивший систему), process id1 (процесс, открывший терминал)

Parent id0, process id2 (k-threat-daemon (kernel)) – демон, который управляет диспетчеризацией (единица - поток).

Размер кучи увеличивается с кол-вом потоков. Многопоточность – большая проблема распределения ресурса памяти. Касается серверов, Яндекс, Гугл… Оптимизация. Вирт память.

Стивен-глава-13.3. Правило прогр демонов.

Ф-ия demonize вызывается из main (main - предок)

1. Сброс маски режима создания файлов: umask(0) – вызывается в предке, в начале demonize.
2. Далее fork() (вызывается только один раз в linux, в System5 – два раза) и завершается процесс предок. (id группы рядом с id предка). Вызвать форк и завершить процесс при этом – в резте потомок – сирота. Для того, чтобы процесс-потомок не был лидером группы. Условие обязательное вызова след функции:
3. setsid(). Осиротевших потомок станет демоном. Станет лидером группы, сессии, но будет один, сам по себе. Утратит управл терминал. Собственный id, сессии id, группы id – особенность демонов. L маленькая в конце функции – многопоточность.
4. chdir(“/”) – изменить директорию, чтобы можно было отмонтировать ФС. В Linux поддерживается большое кол-во файловых систем – EXT2 (родная). В основе лежит монтирование ФС. Флешка монтируется к дереву ФС (у флешки собственная ФС), работать можно только со смонтированной ФС. Подробности в след семе. Системные демоны в состоянии S – интераптабол слип, до тех пор, пока в системе не нужно делать чёт. Пока демон работает, смонтировать ФС не получится.
5. getrlimit(RLIMIT\_NOFILE, &r1), r1- struct rlimit. В этой структуре есть поле r1.rlim\_max, которое возвр макс номер открытого дескриптора, затем, используя эту инфу, сопоставляется с константой RLIM\_INFINITY. [ДОПИСАТЬ]Связан с тем, что наследует потомок от предка. Потомк наследует: код предка, дескрипторы открытых файлов (режим создания файлов установил предок, потомок унаследовал, открытые файлы не нужны. Есть просто файл, инфа о кот мб прочитана, и открытые, создаются специальные системные таблицы. Предок мог открыть любое кол-во файлов, установить права доступа, но всё это не нужно потомку, который стал демоном.), маску создания файлов, сигнальную маску. Потомок унаследует сброшенную маску (чтобы мог создать файлу с доступом. Демонстрация, что потомок наследует маску). Где ещё можно сбросить? В самом потомке, после fork(). (Но тут педагогический приём).
6. fd [знак не сущ] = open(“dev/null”, O\_RDWR). [ДОПИСАТЬ] Предлагает опять открыть эти дескрипторы и отправить в dev/null (тк некоторые демоны открывают 0 1 2 файлы, ну и чтобы их программист не открыл, чтобы не возникло ошибок). Функции sigAction передаётся структура и в поле handler устанавливается значение игнорирования. В конце функции demonize инициализируется файл журнала функциями openlog и выполняется syslog(). Параграф 13.4. При защите демонстрировать журнал ошибок.

alredy\_running следующая функция, выполняющаяся в main (13.5 – Демоны в единственном экземпляре).

Почему у демона не должно быть управляющего терминала? Чтобы нельзя было влиять на его работу. Вы своего демона запускаете из командной строки. Никакого сценария не создаём. Хоть и нет упр терминала, но есть id, kill(-9) – убить своего демона (но не системного, он это игнорирует).

Почему демон должен выполняться в единственном экземпляре? Демон выполняет определённую задачу, зачем нам ещё демоны, выполняющие её, или как её между ними распараллеливать. В книге – файл блокировки.

open() /var/run/<имя> - тут нужны права суперЮзера, директория доступна только ему, чтобы нельзя было легко удалить. Для выполнения остальных действий права такие не нужны.

В open() передаётся LOCKMODE. После этого вызывается lockfile() – функция Стивена. Можно использовать другой системный вызов lockfd. Первый экземпляр демона создаст lockfile – заблокирует его для записи. Следующие попробуют записать в этот файл – не смогут, вернут ошибку.

13.6 Соглашения для демонов. Про var/run

Листинг 13.3 – демон, который перечитывает конфиг файл. [ДОПИСАТЬ]

Два потока: главный и дополнительных. В state увидим многопоточность l.

Упражнения из учебника (3.2, 3.3).

Замечания в лабе:

1. не писать parent finish, мы не знает, финиш он или не финиш
2. массив child, просто обрабатываем
3. parent получит id каждого child (child[i])
4. ничего страшного, если строка будет повторена 2 раза. Даже лучше. Главное видеть – предок создаёт группу.
5. 6 макросов, можно написать функцию.
6. Мануал откройте. Проверяйте на -1.
7. exec(). Что туда передаём. Если есть ввод с клавы, ввод/вывод – блокировка. Процесс будет заблокирован, потеряет квант, др потомок. Это хорошо, что мы увидим, это не ошибка.
8. “привет парент, я чайлд 1” – писать осмысленные фразы – это смешно.
9. Одно сообщение 3а, другое – 25х
10. Пятая программа: писать нормально, с флагом. Копипаст 4ой программы, куда добавляем обработчик сигнала.
11. Проблема: объявлено 2 переменный msg и buf. Педагогический приём. Объявляем только msg. Сам пайп – это buf. Отправили сообщение в трубу – она освободилась.
12. Оптимизация. Что стало лучше? Оптимизация fork() говорит о том, что копируется не весь код предка, а создаются копии необходимых страниц. Без БОЛЕЕ/МЕНЕЕ, лучше ЭФФЕКТИВНО.