**что делает umask? где вызывать? почему? зачем?**

umask -- сброс маски режима создания файлов.   
вызывается в коде предка, потому что

Маска, наследуемая от запускающего процесса, может маскировать некоторые биты прав доступа. Если предполагается, что процесс-демон будет создавать файлы, может потребоваться установка определенных битов прав доступа.

**зачем делать fork() и завершать родительский процесс?**

чтобы гарантировать, что потомок не будет лидером группы -- условие для setsid

Гарантированно созданный процесс не был лидером группы – это условие вызова функции setsid(). Дело в том что когда процесс вызывает fork он становится лидером группы, мы завершаем предка, потомок теряет группу и лидером группы не будет.

**что делает setsid? чем становится процесс? какие условия вызова?**

создает новую сессию, делает процесс лидером новой сессии, лидером новой группы, лишает его управляющего терминала.  
процесс становится процессом-демоном  
условия вызова: процесс не должен быьь лидером группы

Установить session id. Лишает процесс управляющего терминала, делает его лидером группы и лидером сессии, но в этой группе и в этой сессии он один. Это видно по выводу ps

Гарантированно созданный процесс не был лидером группы – это условие вызова функции setsid().

**какая функция обеспечивает демона в единственном экземпляре?**

already\_running

**чем должен обладать демон для создания файла блокировки в /var/run?**

привелегиями superuser

**что за сигнал SIGHUP?**

сигнал, возникающий при разрыве связи процесса с управляющим терминалом

**поток -- единица чего?**

поток -- единица диспетчеризации

**что такое поток?**

Поток -- непрерывная часть кода программы, которая может выполняться параллельно с другими частями кода

**кто владелец ресурсов в системе?**

процесс

**какой процесс рассматривается как поток? как он называется?**

процесс, в котором не создается потоков. он наз. главным потоком

**что делает pthread\_join?**

блокирует процесс, создавший поток, в ожидании его завершения

**что делает pthread\_exit?**

завершает выполнение потока, в котором она была вызвана

**TODO зачем демону утрачивать управляющий терминал?**

**почему демон должен быть в единственном экземпляре?**

Для исключения неоднозначности.  Процесс не может определить, к какому из экземпляров одного и того же демона обращаться; экземпляры не могут взаимодействовать между собой.

**засчет какого механизма получается демон в единственном экземпляре?**

засчет создания файла блокировки

**зачем в демоне закрывать все открытые дескрипторы?**

потому что они унаследованы от предка и являются ненужными

\*\*`ps -ajx`. Столбцы, что они означают?\*\*

STAT=Ssl

S - процесс находится в режиме прерываемого сна.

s - процесс является лидером сессии.

l - процесс является многопоточным.

TTY=? - имя терминал, отвечающего за процесс. ? - нет управляющего.

Tpgid=-1 - идентификатор группы процесса, который в настоящий момент владеет терминалом tty, к которому подключён данный процесс.

PID = pgid = sid - собственный идентификатор pid - идентификатор сессии sid = идентификатор группы pgid

Uid = 0 - user id - привилегии суперпользователя => идентификатор пользователя uid = 0

ps -ajx:

-a информация о процессах

-j инф о pid ppid pgid tty

-x выводит процессы без управляющего терминала

\*\*Какой сон можно прервать?\*\*

Ожидание завершения события

\*\*В какой точке процесс становится демоном?\*\*

При вызове функция demonize

\*\*В какой точке процесс становится многопоточным?\*\*

При вызове функция pthread\_create

\*\*После чего становится лидером?\*\*

После вызова setsid

\*\*Почему демоны должны в единственом системе экземляре?\*\*

Каждый экземпляр делает одно и то же, что делает другой экземпляр.

\*\*Где обеспечивает единственный экзамепляр?\*\*

При вызове функция already\_running

\*\*Какой прием обеспечивает единственный экзамепляр?\*\*

Файл блокировки lockfile

\*\*Почему файл блокировки находится в /var/run/ ?\*\*

По соглашению для демонов

\*\*Как процесс создается?\*\*

Сначала выделяет pid, зачем fd, ...

\*\*Почему такаой ppid?\*\*

Потому что это терминальный процесс

\*\*Почему разработчики подсчитали, что демоны в единственном экземпляре?\*\*

Потому что большее время находится в режиме сна.

=== Русский Язык. Вопросы: ===

**1) Почему у демона нет терминальной группы?**

Ответ: потому что нет управляющего терминала, а значит и терминальной группы

**2) Сколько в программе потоков? Какие?**

Ответ: два: основной и дополнительный

**3) Точка в программе, где демон становится многопоточным?**

Ответ: вызов pthread\_create

**4) Зачем создаем второй поток?**

Ответ: чтобы перехватывать сигналы главного

**5) Какие сигналы перехватывает поток?**

Ответ: SIGHUB и SIGTERM (кста в этом месте, если у вас стоит не switch-case или switch-case без default, то пошлет писать как в Стивенсоне)

**6) Вот вы после создания демона вызываете функцию already\_running - зачем?**

Ответ: чтобы обеспечить работу демона в единственном экземпляре

**7) А конкретнее?**

Ответ: функция сообщает, что демон уже запущен

**8) Какой прием используется для обеспечения единственности экземпляра?**

Ответ: Файл блокировки (lockfile)

**9) Поясните за already\_running**

Ответ: при первом заходе создается файл, при последующих открывается. далее блокируем файл

**10) Построчно объясните как работает lockfile**

Ответ: мы инициализируем экземпляр структуры flock и заполняем поля (кста на другие формулировки она агрится). в type ставим flag\_writelock...

**11) Что определяют ТРИ поля структуры флок?**

Ответ: блокировку на весь файл

**12) Что с этим делаем?**

Ответ: передаем в вызов fcntl

**13) Что делает fcntl?**

Ответ: file control - управление файлами. задает разрешения/режимы работы с файлами

**14) Прокомментируйте main. Каким образом что тут делаете?**

Ответ: инициализируем структуру sigaction, которая описывает поведения при обработки сигналов. Устанавливаем реакцию на реакцию по умолчанию

**15) Что значит по умолчанию?**

Ответ: способом, установленным системой

**16) Зачем делаете sigemptyset?**

Ответ: сбрасывается маска множества/сета сигналов, которые будут заблокированы (sa\_mask).

**17) Дальше...**

Ответ: зануляем специальное поведение (sa\_flags/моя формулировка очень стремная)

**18) в sigaction что делаем?**

Ответ: определяем поведение для sighup

**19) Дальше...**

Ответ: sigfillset - заполняем маску сигналов, которые будут блокироваться. А затем уже определяем поведение для потока.

**20) Для какого потока?**

Ответ: для дополнительного

**21) Что значит SIG\_BLOCK?**

Ответ: это union текущей маски + заполненной маски

**22) Ну а в целом, зачем такие преобразования с масками были нужны?**

Ответ (неуверен): Мы сбросили маску. Сброшенную маску передали в sigfillset. Дальше шла какая-то фраза про "Нет ограничений на прерыв сигналов...". Установили SIGHUP по умолчанию.

**23) То есть что мы сделали?**

Ответ: сначала объявили такое поведение, а затем вызвали поток. название pthread\_sigmask говорит о том, что мы это делаем для потока, вызванного функцией pthread\_create

**24) Почему демон должен существовать в единственном экземпляре?**

Ответ: Каждый экземпляр одного демона делает одно и тоже, что и другой экземпляр того же демона

**25) Нет, вы вообще видели системные демоны? Там, kthreadd, flush... Почему они в единственном экземпляре?**

Ответ: Системные демоны выполняют важные системные функции. В системе решено, какой конкретный экземпляр должен что делать. Система должна определять, какой демон был вызван в какой конкретный момент. Несколько экземпляров одного и того же демона будут работать с одними и теми же ресурсами. Тогда мы вынуждены писать для них взаимоисключение. Как говорится нахуя, если можно реально сделать на каждый цпу по одному демону на каждый ресурс.

**26) У вас код не работает! У вас слово return написано через i! Где r?**

Ответ: (приближаете код) Вот, смотрите: ар, и, ти, ю, АРРРР, эн. (её ответ - шрифт плохой в vscode)

1. Системный вызов функции setsid должен проверяться на -1

2. Зачем нужна функция setsid? Она делает процесс лидером новой группы лидером сессии и лишает его управляющего терминала.

3. Закомментировать в функции daemonize то, что к sighup относится, где мы обеспечиваем невозможность обретения управляющего терминала в будущем. И посмотреть, что будет.

На самом деле никаких различий не будет, процесс не завершится, он будет также работать. Правильный ответ я не поняла, что-то связано с тем, что сигнал рассылается всем процессам.

4. Объяснить весь код функции main, прям посрочно, после already\_running и до pthread\_create.

Для сигнала SIGHUP восстанавливается действие по умолчанию (SIG\_DFL). Сбрасывается маска. Устанавливаются флаги для структуры sa: 0 значит, что не используются никакие специальные флаги для обработки сигнала. Функция sigfillset заполняет набор сигналов mask всеми доступными сигналами.

Дальше SIG\_BLOCK-это union, объединение. Про блокировку здесь говорить нельзя, она считает это неправильным. Как сказать, что именно тут делается, я не знаю. Но объединение делается для того, чтобы сигналы доставлялись не этому потоку, а тому, который мы создаём в pthread\_create.

5. В файл конфигурации должно записаться не root, а прям имя пользователи, там просто функция getlogin().

6. У кого-то из ребят просила объяснить функции already\_running и lockfile, построчно.