Процессы UNIX

UNIX является многозадачной операционной системой. Это означает, что одновременно может быть запущена более чем одна программа. Каждая программа, работающая в некоторый момент времени, называется процессом. Каждая команда, которую вы запускаете, порождает хотя бы один процесс. Есть несколько системных процессов, запущенных все время и поддерживающих функциональность системы.

У каждого процесса есть уникальный номер, называемый process ID, или PID, и, как и у файлов, у каждого процесса есть владелец и группа. Информация о владельце и группе процесса используется для определения того, какие файлы и устройства могут быть открыты процессом с учетом прав на файлы, о которых говорилось ранее. Также у большинства процессов есть родительский процесс. Например, при запуске команд из оболочки, оболочка является процессом и любая запущенная команда также является процессом. Для каждого запущенного таким путем процесса оболочка будет являться родительским процессом. Исключением из этого правила является специальный процесс, называемый init. init всегда первый процесс, его PID всегда равен 1. init запускается автоматически ядром во время загрузки.

Жизненный цикл процесса

Для запуска программ в UNIX используются два системных вызова – fork() и exec().

fork() — системный вызов, создающий новый (дочерний) процесс, идентичный выполняющему этот вызов. После вызова fork() алгоритм обычно разветвляется (родительский процесс получает от fork() значение PID дочернего процесса, а дочерний получает нуль).

После fork() дочерний процесс чаще всего выполняет системный вызов exec(), загружающий в пространство процесса новую программу (именно так, и только так, в UNIX выполняется запуск программы в отдельном процессе). Так, первый (нулевой) процесс UNIX (ядро системы) создаёт свою копию, чтобы запустить init (процесс с PID = 1), который в свою очередь создаёт дочерние процессы для запуска инициализации системы и терминалов.

При завершении работы программа (дочерний процесс) выполняет системный вызов exit(), при этом родительский процесс с помощью системного вызова wait() (или waitpid()) должен очистить таблицы планировщика процессов от информации о своем дочернем процессе. Если этого не происходит и родительский процесс завершается, не вызвав wait() для всех своих дочерних процессов, в системе возникают зомби (zombie), представляющие собой записи в таблицах планировщика процессов. Очистить операционную систему от зомби можно только с помощью перезагрузки.

Запуск команд в консоли в фоновом режиме

Для запуска заданий в фоновом режиме используется специальный символ &, указывающий шеллу, что запускаемая программа должна выполняться на заднем плане (background):

```
$ find / -name '*.gz' &
 $ opera &
```

Типы процессов

Процессы в UNIX можно разделить на следующие типы:

- Системные (vmdaemon, pagezero, bufdaemon, syncer)
- Демоны/сервисы (usbd, httpd, sshd)
- Интерактивные/прикладные процессы (ls, sh, fsck …)
- Процесс init

Команды мониторинга процессов

Две команды очень полезны для просмотра работающих в системе процессов, это ps(process status, показывает моментальный снимок процессов в системе) и top(table of processes, позволяет просматривать информацию о процессах в реальном времени). Команда ps используется для получения списка запущенных процессов и может показать их PID, сколько памяти они используют, команду, которой они были запущены и т.д. Команда top показывает запущенные процессы и обновляет экран каждые несколько секунд, что позволяет наблюдать за работой компьютера в реальном времени.

Подробную информацию об этих программам можно получить на соответствующих страницах справки (man ps и man top).

Атрибуты процесса

- PID – Идентификатор процесса
- PPID (ключ j в ps) – Идентификатор родительского процесса
- TTY (столбец TT в ps) – Контрольный (управляющий) терминал
- RUID, EUID (ключи alw в ps) – Реальный и эффективный (действующий) UID
- RGID, EGID (ключи alw в ps) – Реальный и эффективный (действующий) GID

Базовые механизмы взаимодействия программ в UNIX

- Перенаправление потоков ввода/вывода
- Переменные окружения
- Коды завершения
- Сигналы
- Межпроцессные взаимодействия (IPC – InterProcess Communications):
- unix- и сетевые сокеты
- разделяемая память
- именованные каналы

Система безопасности UNIX

Многопользовательская среда предполагает наличие механизма регулирования прав доступа к любому ресурсу в системе. Существует три типа прав доступа: на чтение, запись и исполнение. Права сгруппированы три по три, соответственно чтение/запись/выполнение для владельца/группы/всех остальных. Численное представление:

Значение	Права доступа	Символьное представление
0	Ничего не разрешено	---
1	Нельзя читать и писать, разрешено исполнять	--x
2	Нельзя читать и исполнять, разрешено писать	-w-
3	Нельзя читать, разрешено писать и исполнять	-wx
4	Разрешено читать, нельзя писать и исполнять	r--
5	Разрешено читать и исполнять, нельзя писать	r-x
6	Разрешено читать и писать, нельзя исполнять	rw-
7	Разрешено все	rwx

Опция -l команды ls используется для получения подробного листинга каталога,

включающего колонку с информацией о правах на файл для владельца, группы и всех остальных. Например, команда ls -l в произвольном каталоге может вывести следующее:

```
% ls -l
 total 530
 -rw-r--r-- 1 root wheel 512 Sep 5 12:31 myfile
 -rw-r--r-- 1 root wheel 512 Sep 5 12:31 otherfile
 -rw-r--r-- 1 root wheel 7680 Sep 5 12:31 email.txt
 ...
```

Вот как выглядит первая колонка вывода ls -l:

-rw-r--r--

Первый (считая слева) символ говорит обычный ли это файл, каталог, символьное устройство, сокет или любое другое псевдо-файловое устройство. В нашем случае - указывает на обычный файл. Следующие три символа (в данном случае это rw) задают права доступа владельца файла. Затем идут права группы, которой принадлежит файл (r). Последняя тройка (r) определяет права для всех остальных. Минус означает отсутствие каких-либо прав (т.е. нельзя ни читать, ни писать, ни выполнять). В данном случае права установлены таким образом, что владелец может читать и писать в файл, а группа и другие могут только читать. Таким образом, численное представление прав 644, где каждая цифра представляет три части прав на файл.

Права на устройства контролируются аналогичным образом. В UNIX все устройства представлены в виде файлов, которые можно открывать, читать и писать в них. Эти специальные файлы содержатся в каталоге /dev.

Каталоги также являются файлами. К ним применимы те же права на чтение, запись и выполнение. Правда, в данном случае 'выполнение' имеет несколько другой смысл.

Когда каталог помечен как 'исполнимый', это означает, что можно 'зайти' в него (с помощью команды cd, change directory). Это также означает, что в данном каталоге можно получить доступ к файлам, имена которых известны (конечно, если собственные права на файл разрешают такой доступ).

Если же требуется получить список файлов в некотором каталоге, права доступа на него должные включать доступ на чтение. Для того, чтобы удалить из каталога какой-либо файл, имя которого известно, на этот каталог должны быть даны права на запись и на исполнение. Существуют и другие права доступа, но они как правило используются в особых случаях, например, SUID (Set UID) и SGID (Set GID) -бит на выполняемые файлы и sticky-бит на каталоги.

Символические обозначения прав

Символические обозначения, иногда называемые символическими выражениями, используют буквы вместо восьмеричных значений для назначения прав на файлы и каталоги. Символические выражения используют синтаксис (кто) (действие) (права), где существуют следующие значения:

(кто) u Пользователь (User)
(кто) g Группа (Group)
(кто) o Другие (Other)
(кто) a Все (All, 'world')
(действие) + Добавление прав
(действие) - Удаление прав
(действие) = Явная установка прав
(права) r Чтение (Read)
(права) w Запись (Write)
(права) x Выполнение (Execute)
(права) t Sticky бит
(права) s SUID или SGID

Эти значения используются командой chmod(1) так же как и раньше, но с буквами. Например, вы можете использовать следующую команду для запрета доступа других пользователей к FILE:

% chmod go= FILE

Для изменения более чем одного набора прав можно применить список, разделенный запятыми. Например, следующая команда удалит права группы и 'всех остальных' на запись в FILE, а затем добавит права на выполнение для всех:

% chmod go-w,a+x FILE

Управление маской доступа

Команда umask

Для управления правами доступа вновь создаваемых файлов используется команда umask. Запущенная без параметров, она показывает текущую установленную маску, а с параметром – устанавливает указанное значение. Биты, присутствующие в маске, будут отсутствовать (будут замаскированы) в правах доступа созданного файла.

Команда chmod

```
[hostX:~] # chmod 640 file1
 [hostX:~] #
 [hostX:~] # chmod o-r file1
```

Управление атрибутами владельцев файла

Команда chown

Команда chown меняет владельца файла. Только суперпользователь (UID=0) может использовать эту команду.

Команда chgrp

Как следует из названия, команда chgrp позволяет изменять группу файлов. При этом пользователь должен принадлежать целевой группе и быть владельцем файла или же быть суперпользователем.

Повышение привилегий пользователей

Команды с setuid битом

$ passwd

Команда su

```
[hostX:~] # pw usermod uX –G wheel
 [hostX:~] # cat /etc/group
 ...
 wheel:*:0:root,uX
```

Подключаемся непривилегированным пользователем входящим в группу wheel

$ su

Пакет sudo

```
[hostX:~] # pkg_add /usr/ports/packages/All/sudo.tbz
 [hostX:~] # visudo
 ...
 %wheel ALL = (ALL) ALL
 userX ALL = NOPASSWD: mount /cdrom
 ...
```