Создание среды окружения

Окружение (environment) или среда набор пар ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ, доступный каждому пользовательскому процессу.

Окружение - набор *переменных окружения*.

Переменные окружения (п.о.) специальные переменные, определенные оболочкой и используемые программами во время выполнения.

Типы п.о.:

1) Локальные
(определены только для текущей сессии)**;**
2) Пользовательские
(определяются для конкретного пользователя)**;**
3) Системные
(доступны во всей системе для всех пользователей)

Команда **env -** посмотреть окружение.
$env –i bash – запуск оболочки без п.о.

Создание локальной п.о.:
$var=значение или **$export var=значение**;

$echo $var – посмотреть значение п.о. **var**;

$unset var – удалить п.о. **var**.

Пример:
$PS1='Yes, dear ?‘ –
задание строки-приглашения PS1.

$echo $PS1 – посмотреть значение п.о. **PS1**;

$unset PS1 – удалить п.о. **PS1**.

Команды и переменные, заданные в **.bash_profile (.bashrc)**, сохраняются при завершении сессии.

$source .bash_profile – применить изменения в **.bash_profile**.

Интерпретатор выделяет следующие п. о.:

PS1 – шаблон строки-приглашения;

PS2 – вторичное приглашение;

HOME - имя домашнего каталога;

Пример: **$echo $HOME**

/home/A52_1

PATH - путь для поиска команд по умолчанию.

Пример: **PATH=.:/bin:/usr/bin**

TERM - тип используемого терминала.

MAIL - имя стандартного файла, в котором хранится почта.

Пример: **MAIL=/usr/spool/mail/A52_1.**

Можно применять п. о. для сокращения записи:

d = /very/long/directory/name

Тогда

$cd $d - переход в директорию **name**.

Чтобы п. о. были доступны всем процессам оболочки, используют команду

export

$export MAIL PATH TERM d

Архивация в Unix

Архивирование - объединение нескольких небольших файлов в один с целю последующей передачи, хранения, шифрования или сжатия.

tar - наиболее распространенный архиватор, апримeняемый в Linux-системах. Не использует сжатие.

Для сжатия используют другие утилиты, например, **gzip** или **bzip2**.

Поэтому эти программы используются вместе.

tar создает несжатый архив, в который помещаются выбранные файлы и каталоги.

Полученный файл *.tar сжимается архиватором, например, gzip.

Параметры архиватора tar

-v - подробный вывод информации (о размере, правах и проч.) и работе;

-f - запись в файл (указывается всегда при создании и распаковке архива);

-c - создание нового архива (исп. вместе с **-f**);

-A – добавить файлы в несжатый архив;

-r - добавить файлы в конец архива;

-d - показать различия между архивами;

-t - вывод списка файлов в архиве;

Параметры архиватора tar

-u - добавление в архив файлов, новее уже существующих в архиве;

-x - извлечь файлы из архива;

-j - указание типа архива bzip2 (сжатие);

-z - указание типа архива gzip (сжатие);

-k - не перезаписывать существующие файлы;

-w – интерактивный режим (запрос подтверждения действий);

-X имя_файла – не добавлять в архив файлы, указанные в списке «имя_файла».

Использование архиватора tar

tar запускается с обязательным указанием одного из основных действий - создание или распаковка архивов.

Несжатые архивы имеют расширение **.tar**.

Пример 1:

$tar -cf txt.tar *.txt

Упаковка всех файлов с расширением .txt в архив **txt.tar**.

Пример 2:

$tar -czf files.tar.gz ~/files

упаковка папки ~/files со всем содержимым в сжатый с помощью gzip архив files.tar.gz.

Использование архиватора tar

Пример 3:

$tar –czf a.tar.gz `find / -name “a*” –type f`

упаковка всех файлов системы, имя которых начинается на «a», в сжатый архив a.tar.gz.

Пример 4:

$tar -xf /path/to/archive.tar.bz2

распакует содержимое архива в текущую папку.

Пример 5:

$tar -xvf archive.tar.bz2 -C /path/to/folder

распаковка архива в папку /path/to/folder

Пример 6:

$tar -tvf archive.tar.gz

просмотреть содержимое архива.

Процессы в Unix

Процесс - программа в стадии ее выполнения.

Процесс включает в себя:

- программный код – последовательность команд, исполняемых процессором;
- данные, обрабатываемые программой;
- стек – динамически выделяемая память для хранения оперативной информации;
- системную информацию о выполняемой задаче (размещаемая память, открытые файлы, статус процесса, системные переменные окружения).

Атрибуты процессов

- PID – уникальный целочисленный идентификатор процесса;
- PPID – идентификатор родительского процесса (процесса, породившего данный);
- Приоритет процесса – число в пределах от –20 до 20, определяющее относительную долю процессорного времени, отводимого данному процессу. Значение –20 соответствует максимальному приоритету, 0 – стандартному приоритету;
- TTY – терминальная линия, т.е. устройство, с которым по умолчанию ассоциирован стандартный ввод и вывод
- RID и RGID – соответственно идентификаторы пользователя, запустившего процесс, и его группы.
- EID и EGID – то же, но определяют пользователя, от имени которого действует процесс, что задает права доступа к файлам и системным операциям. Обычно (но не всегда) совпадают с RID и RGID.

Атрибуты процессов выводятся командой ps.

Состояния процессов

Взаимодействие процессов

Взаимодействие процессов в системе осуществляется посредством:

- передачи данных между процессами;
- совместного использования данных;
- извещения о наступлении каких-либо событий.

Средства взаимодействия между процессами:

- сигналы (средство прерывания работы процесса);
- каналы (pipe);
- именованные каналы (однонаправл. средство п.д.);
- сообщения;
- семафоры;
- разделимая память;
- сокеты.

Процесс может выбрать одно из трех возможных действий при получении сигнала:

- *игнорировать сигнал*;
- *перехватить и самостоятельно обрабатывать его*;
- *позволить действие по умолчанию*.

Средством посылки сигнала служит команда

kill -№sign PID

Пример: послать с терминала сигнал завершения процессу, который запущен в фоновом режиме:

```
$ ./long-program &
$ kill $!
```

($! содержит PID последнего процесса, запущенного в фоне)

По умолчанию **kill** посылает **SIGTERM**.

Иногда процесс продолжает существовать при получении SIGTERM.

Тогда к нему применяется более жёсткое средство – сигнал **SIGKILL**, который нельзя ни перехватить, ни игнорировать:

$kill -9 PID или
$kill -SIGKILL PID

Уничтожить все процессы за исключением начального процесса-интерпретатора:

$ kill 0

Уничтожить все процессы с именем **proc**:

$ killall proc

Пример:
$ killall –SIGKILL top
принудительно завершить процесс top.

Команды управления процессами

ps – вывод информации о запущенных процессах.

Ключи: **a** - процессы, связанные с текущим терминалом, и процессы других пользователей;

x - процессы, отсоединённые от терминала (демоны, службы);

aux - вывод всех процессов в системе;

-ejH - вывод процессов в виде дерева;

-u user - процессы пользователя user;

-o format – вывод информации в определённом формате.

Команда отображает информацию в виде
PID TTY TIME CMD

$ nohup команда

команда продолжает выполняться после выхода из системы. Результат выполнения команды сохранится в файле **nohup.out**.

$ nice -n value команда

изменение приоритета запускаемого процесса *команда* на значение, равное *value* (может быть от -20 до 19, в порядке уменьшения приоритета).

$ renice -value PID

изменение приоритета запущенного процесса с PID на значение, равное value.

Пример:

$ renice +5 1248 – понижение приоритета процесса с номером 1248 на 5.

Чтобы запустить процесс в заданное время, используется команда:

$ at время
любые команды…
ctl-d

или из файла:

$ at 3am << файл

$ jobs - список остановленных и фоновых задач.

$ bg number - продолжить выполнение остановленной задачи в фоне, имеющей номер number.

$ fg number – перевести задачу с номером *number* из фонового режима в активный.

$ top

показать все запущенные процессы в интерактивном режиме.

Клавиши для управления командой:

h - справка о программе;

k - уничтожить процесс;

n - число отображаемых процессов;

u - сортировать по имени пользователя;

M - сортировать по объему ОЗУ;

P - сортировать по загрузке ЦП;

r - изменить приоритет выполнения;

q – выход.