ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 5.
«УПРАВЛЕНИЕ ПРОЦЕССАМИ. ДОСТУП ПРОЦЕССОВ К ФАЙЛОВОЙ СИСТЕМЕ»

Теоретическая часть

Под процессами во всех Unix-подобных системах подразумевается выполняющаяся программа и её реальные и потенциальные взаимосвязи с частью ресурсов исполняющего её компьютера. Процесс, как единица вычислительной работы, управляемой ОС, тесно связан с такими свойствами управляемой вычислительной системы, как учетные записи пользователей, права доступа пользователей (опосредованно и процессов) и файловая система.

Связь процессов с ресурсами компьютера обладает свойствами:

- пользователь (реальный или виртуальный) запускает процессы, порождающие файлы;
- права доступа процесса соответствуют правам доступа запустившего его пользователя;
- порожденные процессом файлы получают права, соответствующие правам процесса;
- права пользователя определены параметрами его учетной записи.

Каждый процесс уникален даже при совпадении запускающей его команды и программы. Для однозначной идентификации процесса используется числовое значение - идентификатор процесса (PID, Process Identificator). Для каждого процесса известен владелец
(пользователь, запустивший процесс). Если процесс запущен реальным пользователем, то он привязан к терминалу (многотерминальные или системы с оконным графическим интерфейсом), в котором была дана команда на его запуск. Для виртуальных (системных) пользователей такой ассоциации не предусмотрено. Также каждый процесс связан с родительским процессом по его идентификатору (PPID, Parent PID, запросите вывод команды pstree). В каждый момент времени операционная система учитывает и предсказывает состояние процесса - степень его исполнения или возможность продолжения. Процесс может быть:

- выполняемым в текущий момент (R, Runned);
- находящимся в режиме ожидания (S, Suspended);
- прерванным (T, Terminated), например, при использовании клавиш Ctrl+Z;
- "зомби" (Z, Zombied) такой процесс, который завершился, но родительский процесс еще не получил сигнала его завершения. Спустя некоторое время "зомби" завершаются окончательно и освобождают ресурсы;
- зависшим, или в состоянии непрерывного ожидания (uninterruptible sleep). Такой процесс не реагирует на какие-либо сигналы и может быть окончательно завершён только завершением работы операционной и компьютерной системы.

Еще одной характеристикой процесса (помимо формального состояния) является уровень приоритета (NI, NIce value, "степень дружественности"). Он влияет на количество системных ресурсов (интенсивность выделения), выделяемых процессу.

Основными командами для получения сведений о выполняемых процессах являются ps и top. Фрагмент вывода сведений командой ps с параметрами a (расширенный вывод), u (с указанием UID), x (в т.ч для виртуальных пользователей):

```
user@VirtualBox:~> ps aux
USER        PID %CPU %MEM VSZ   RSS   TTY   STAT START  TIME COMMAND
COMMAND
root        2   0.0  0.0  0     0     ?     S    янв29  0:00 [kthreadd]
root        3   0.0  0.0  0     0     ?     I<   янв29  0:00 [rcu_gp]
root        4   0.0  0.0  0     0     ?     I<   янв29  0:00 [rcu_par_gp]
```

```
root         5  0.0  0.0      0     0 ?        I<   янв29   0:00 [slub_flushwq]
root         6  0.0  0.0      0     0 ?        I<   янв29   0:00 [netns]
root        25  0.0  0.0      0     0 ?        S    янв29   0:00 [kdevtmpfs]
...
root        92  0.0  0.0      0     0 ?        S    янв29   0:45 [kswapd0]
...
root       414  0.0  0.4  85404 18320 ?        S<s  янв29   0:11 /lib/systemd/systemd-
journald
root      1682  0.0  0.0      0     0 ?        S<   14:28   0:01 [ipw2200/0]
root       799  0.0  0.0      0     0 ?        S<   янв29   0:00 [pktcdvd0]
root      1348  0.0  0.0  12184  2832 ?        Ss   янв29   0:00 sshd: /usr/sbin/sshd -
D [listener] 0 of 10-100 startups
root      3041  0.0  0.1  87596  7776 ?        Ss   янв29   0:00 /usr/sbin/smbd --
foreground --no-process-group
gerbera   1266  0.0  0.2 788192 10920 ?        Slsl янв29   0:03 /usr/bin/gerbera -c
/etc/gerbera/config.xml
postfix   3508  0.0  0.0  38396  3140 ?        S    янв29   0:00 qmgr -l -t unix -u
proftpd   3621  0.0  0.0  21196  1988 ?        Sls  янв29   0:03 proftpd: (accepting
connections)
```

Для удобства расширяемости сервисные программы ОС хранятся в файлах имена которых совпадают к командами даваемыми пользователем-администратором на запуск программ. Вызов системных программ-утилит реализует командный интерпретатор Shell. Большей гибкостью и универсальностью по сравнению с программой ps (вызов командой ps) обладает программа top (вызов командой top). Она позволяет не только получить информацию о процессах, но и выполнять мониторинг через заданные интервалы времени. Так же эта программа позволяет управлять процессами, объединяя функции встроенных в оболочку подпрограмм jobs, nice, fg и kill (вызываются также по имени).

Управление процессами

Пользователь может управлять (доступом процессов к процессору) только теми процессами, владельцем (и инициатором) которых он является, процессы порождаются в процессе ввода-чтения, интерпретации и выполнения команд командной оболочкой Shell. Суперпользователь (root, UID=0) может управлять всеми процессами, а точнее теми процессами идентификатор владельца которых больше или равен 0.

Управление запущенными процессами ограничивается приостановкой выполнения процесса, изменению приоритета при распределении процессорного времени и принудительному (внешнему, безусловному) завершению.

Приостановить выполнение активного процесса, использующего терминал для вывода информации, можно командой прерывания, связанной в терминале сочетанием клавиш Ctrl+Z. Для продолжения работы процесса после его прерывания дают команду fg. Если имеется несколько приостановленных процессов, то в качестве параметра команды fg необходимо указать порядковый номер задания в текущей оболочке, (не путать с PID), работу которого нужно продолжить. Узнать номер задания и связь его с командами активизации процесса можно командным запросом Shell jobs.

Изменение приоритета (при использовании процессора в режиме разделения времени) отдельных процессов возникает при необходимости перераспределения ресурсов системы. Значения уровня приоритета (nice value) изменяется от -20 (наименьшая "дружественность" к окружающим процессам, высший приоритет) до +20 (низший приоритет, снижение приоритета использования процессора и как следствие уменьшение частоты использования внешних устройств). Все пользовательские (и большинство системных) процессы запускаются с равным и одинаковым начальным приоритетом (nice value = 0). Для понижения приоритета ранее запущенной задачи используется команда renice с указанием уровня и PID задачи:

```
[aag@localhost ~]$ renice --7 20117.
```
Команду nice целесообразно использовать тогда когда нужно запустить процесс с заранее заданным приоритетом. Формат написания команды `nice <значение приоритета> <имя процесса>`. Команда `sudo renice +20 -u 1000` понижает приоритет всем процессам пользователя с идентификатором 1000 (см. вывод команды id). То же делает команда `sudo renice +20 -g 1000` в отношении всех процессов заданной группы пользователей (см. вывод команды `groups` и значение параметра `gid` в выводе команды `id`).

Пользователь имеет право понижать приоритет собственных задач, т. е. Тех процессов которые запущены по его командам. Повышать уровень приоритета любой задачи может только суперпользователь. Гораздо чаще, чем изменение приоритета, возникает необходимость принудительного завершения (снятия) процесса. Такая ситуация возникает, например, когда процесс "зависает", то есть перестает воспринимать нажатия клавиш и не отвечает на системные события, но продолжает использовать и удерживать компьютерные ресурсы. Для снятия с выполнения "зависшей" программы администратор даёт команду kill, которая посылает процессу один из сигналов завершения. Список сигналов доступен по команде-запросу kill -l, а их подробное описание - по команде man 7 signal. Здесь же отметим, по умолчанию типа сигнала, процессу будет передан сигнал SIGTERM (номер 15), предписывающий по возможности корректно, с сохранением информации из оперативной памяти на энергонезависимом устройстве хранения, завершить работу.

Примеры использования команды: вызов со значением сигнала по умолчанию (SIGTERM)

```
[aag@localhost ~]$ find / *.html
[aag@localhost ~]$ ps
PID    TTY    TIME       CMD
11425  pts/0  00:00:00   bash
52991  pts/0  00:00:08   mplayer
54279  pts/0  00:00:00   bash
54283  pts/0  00:00:00   oosplash
54317  pts/0  00:01:51   soffice.bin
59463  pts/0  00:00:00   ps
20712  pts/1  00:00:00   find
[aag@localhost ~]$ kill 20712
```
Явное указание номера сигнала:
```
[aag@localhost ~]$ kill -15 20712
```
Явное указание имени сигнала (номер 9, SIGKILL, требующий немедленного завершения работы программы):
```
[aag@localhost ~]$ kill -SIGKILL 20712
```
Проанализируем и попытаемся понять взаимосвязь процессов, команд и интерпретации. Снова дадим команду `find / *.html` и как в предыдущем примере приостановим незакончившийся процесс. Найдём PID выполняющейся оболочки командой `ps` и запросим `pstree 11425`. Получим схематичное изображение «дерева» порождения процессов:

```
bash─┬─bash───oosplash─┬─soffice.bin───20*[{soffice.bin}]
     │                 └─{oosplash}
     ├─find
     ├─mplayer─┬─mplayer
     │         └─{mplayer}
     └─pstree
```

Проанализируем, какими системными и пользовательскими процессами занята сейчас компьютерная система. Запустим программу htop и изучим пользовательское (функциональное) меню:

Аналогичное действие для всех процессов запущенных этой командой выражается командой killall find. Если нужно указать точно родительский процесс или любые другие более конкретные условия поиска останавливаемого процесса изучите справку по pkill. Например команда pkill -P 11425 find остановит все процессы find, запущенные пользователем (опосредовано интерпретатором команд) через процесс класса Shell с PID= 11425 (номер выясняется по запросу ps) .

Длительные процессы можно запускать в фоновом режиме. Текущий процесс приостанавливается и переводиться в фоновый режим командой bg. Запланировать сразу запуск в фоновом режиме можно добавив амперсанд в конце строки:

```
find . -name *iso > /tmp/res.txt &
mplayer
http://europaplus.hostingradio.ru:8014/radiosept320.mp3 >
/dev/null 2>&1
```

в первом случае стандартный вывод перенаправляется в файл с заданным именем, а во втором в виртуальное устройство `/dev/null` которое не меняет своего состояния ни при записи ни при чтении. Можно запустить много фоновых процессов. Просмотра идентификаторов и команд всех фоновых процессов текущей оболочки просмотрим командой `jobs`. Перевести нужный процесс из фонового режима в обычный (режим переднего плана) делаем командой `fg` <номер фоновой или приостановленной задачи>.

Команда ipcs сообщает о состоянии средств межпроцессного взаимодействия. (Общая память, семафор и очередь сообщений). Это очень тонкие аспекты администрирования, которые могут быть полезны если прикладная программа не работает, а изучение подробной документации к ней недоступно Опциями-параметрами -p <PID> в сочетании с -m (Shared memory, VRAM), -s (Семафоры) или -q (Очереди сообщений) для получения идентификатора процесса соответствующего межпроцессного взаимодействия.

Бывают ситуации в которых нужно изучать статистику использования ресурсов и объёмы памяти, которую ОС выделяет на управление различными устройствами как физическими так и виртуальными. Запрос к ОС может быть записан так ulimit -a . Проанализируйте информацию, какие из перечисленных параметров влияют на работоспособность процесса и вам известна теоретическая взаимосвязь. В многопользовательских системах важно понимать, какие действия пользователей могли привести к проблемам, решать эту задачу начинают с запроса информации о вошедших в систему пользователях и процессах, которые они выполняют. Дают команду «w» и изучают её вывод. Команда pgrep -u user sh сканирует текущий запущенные процесса, а затем выводит номера процессов на стандартный вывод в соответствии с условиями сопоставления команды. В данном случае «все процессы sh у пользователя user».
Найдём процесс find (F3) и завершим его (F9).

Задания для самостоятельной работы

1. Войти в систему с собственной учетной записью.
2. Получить справку о команде ps.
3. Командой ps вывести краткую информацию о выполняющихся процессах в текущем терминале и определить PID текущей оболочки.

4. Получить подробную информацию о загруженных процессах и выяснить, какой из них использует максимальный объем памяти, а какой - максимально загружает процессор.
5. Из таблицы, полученной в п.4, выяснить, какой PID имеет процесс init и от чьего имени он запущен.
6. Открыть новый сеанс с собственной учетной записью в tty2 и запустить в нем файловый менеджер MC.
7. Вернуться в tty1 и снова просмотреть список процессов.
Определить PID MC, запущенного от вашего имени.
8. Повторить п.6 для пользователей root и stud соответственно в tty3 и tty4.
9. Вернуться в tty1 и определить PID MC, запущенного от имени root и stud.
10.Командой kill снять все процессы MC.
11.Перейти в tty3 (сеанс root) и повторить п.10. Чем можно объяснить различия в результатах выполнения?
12.В tty1 выполнить команду top. Сравнить ее возможности с возможностями ps.
13.Используя top или ps определить, какие процессы порождены (поле PPID) процессом init (PID=1).
14.Завершить сеансы в tty3 и tty4.
15.В tty1 запустить поиск всех файлов .html от каталога /.
Приостановить этот процесс (Ctrl+Z). Запустить команду man bash и приостановить ее выполнение.
16.Командой jobs определить номера задач, запущенных в п. 15.
17.Командой fg продолжить выполнение man bash.
18.Принудительно (kill) завершить команду find.
19.Завершить все открытые сеансы.

Контрольные вопросы
1. Что подразумевается под процессами в Unix-подобных системах?
2. Каким может быть процесс?
3. На что влияет уровень приоритета процесса?
4. Основными командами для получения сведений о выполняемых процессах являются…
5. Что такое изменение приоритета процесса?