Лабораторная работа:
«Команды получения сведений о системе и процессах»

Задание: Вывести результат выполнения следующих команд в терминале ОС Ubuntu

Команды получения сведений о системе

date

Программа позволяет посмотреть текущую дату и время в одном из множества форматов. Суперпользователь также может использовать эту программу для установки текущей даты и времени.
По умолчанию программа выводит время в формате заданной временной зоны. Для получения времени по Гринвичу используется параметр `-u`.
При выводе даты и времени используется формат вывода, принятый для данной страны и задаваемый переменными окружения `LC_TIME` и т.п..
Рассмотрим примеры использования команды:

Текущее время:

```
user@desktop test $ date
Срд Окт 19 08:21:38 MSD 2005
```

Текущее время по Гринвичу:

```
user@desktop test $ date -u
Срд Окт 19 04:21:39 UTC 2005
```

cal

Команда предназначена для вывода календаря на месяц или на год. По умолчанию, выводит календарь текущего месяца.
В качестве параметра можно указать год или месяцы год. Обратим внимание, что команда **cal 05** выведет календарь на 5-й год, а не на 2005-й.
При выводе календаря используется формат вывода, принятый для данной страны и задаваемый переменными окружения `LC_TIME` и т.п..
Рассмотрим примеры использования команды:
Календарь на текущий месяц

```
user@desktop test $ cal
    Октября 2005
Вс Пн Вт Ср Чт Пт Сб
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
      30 31
```

Календарь на 2005 год.

```
user@desktop test $ cal 2005
                                2005
         Января                   Февраля                   Марта
Вс Пн Вт Ср Чт Пт Сб   Вс Пн Вт Ср Чт Пт Сб   Вс Пн Вт Ср Чт Пт Сб
                   1            1  2  3  4  5            1  2  3  4  5
 2  3  4  5  6  7  8    6  7  8  9 10 11 12    6  7  8  9 10 11 12
 9 10 11 12 13 14 15   13 14 15 16 17 18 19   13 14 15 16 17 18 19
16 17 18 19 20 21 22   20 21 22 23 24 25 26   20 21 22 23 24 25 26
23 24 25 26 27 28 29   27 28                  27 28 29 30 31
30 31
```

. . .

ps

Команда *ps* выводит различную информацию о запущенных процессах. Запущенная без ключей, эта команда выводит сводку процессов, связанных с терминалом, с которого ее запустили.
Ключи позволяют:
а) выбрать процессы, информацию о которых следует вывести;
б) указать, какую информацию о процессах выводить.

Основные ключи команды *ps*:
-e вывести информацию обо всех запущенных процессах;
-u *пользователь* вывести информацию о процессах указанного пользователя;
-f "полный" листинг (см. таблицу ниже);
-l "длинный" листинг (см. таблицу ниже);
-j вывести индентификаторы группы процессов и сеанса.

Поля вывода команды ps

Поле	Описание	Ключи*
S	Состояние процесса: **O** - выполняется (*On processor*), **R** - готов к запуску (*Runnable*), **S** - находится в состоянии сна (*Sleeping*), **Z** - зомби (*Zombie*), **T** - остановлен (*Stopped*).	**l**
UID	Идентификатор пользователя, от имени которого запущен процесс (с ключом **-f** выводится имя пользователя)	**f,l**
PID	Идентификатор процесса	*все*
PPID	Идентификатор родительскогопроцесса	**f,l**
PGID	Идентификатор группы процессов	**j**
SID	Идентификатор сеанса	**j**
PRI	Приоритет процесса (чем больше, тем ниже)	**l**
NI	Относительный приоритет (Nice Number)	**l**
SZ	Размер процесса в страницах (размер страницы можно узнать командой *pagesize*)	**l**
STIME	Время запуска процесса	**f**
TTY	Управляющий терминал ('?' - для демонов)	*все*
TIME	Суммарное время, затраченное процессором на исполнение процесса	*все*
CMD	Имя процесса (с ключом **-f** выводятся первые 80 символов командной строки)	*все*

*) - в колонке *Ключи* указано, какой ключ надо дать команде *ps*, чтобы соответствующее поле появилось в выводе. Пометка "*все*" обозначает, что поле выводится всегда, в том числе и при запуске команды без ключей. Ключи **-f, -l, -j** можно использовать совместно для получения комбинированного вывода.
Ключи **-f, -l, -j** не определяют, о каких процессах выводить данные , а устанавливают только формат вывода. Для отбора процессов используйте ключи **-e, -u**.
Команда *ps* имеет также ключ **-o** (буква "o"), параметром которого является список полей вывода через запятую. Таким образом можно выбрать только необходимые поля, а также вывести дополнительные данные о процессе, не перечисленные в таблице выше. Наименования полей для ключа **-o** см. в справочнике *man*.

Для просмотра всех процессов можно воспользуемся следующей командой:

```
user@desktop ~ $ ps aux
```

```
USER     PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       1  0.0  0.0   1432   480 ?        S    13:16   0:01 init [3]
root       2  0.0  0.0      0     0 ?        SN   13:16   0:00 [ksoftirqd/0]
root       3  0.0  0.0      0     0 ?        S<   13:16   0:00 [events/0]
..
user    8456  0.3  4.4  34932 22944 ?        S    13:24   2:15 emacs
user   22537  0.0  0.3   3720  1560 pts/3    Ss   22:37   0:00 /bin/bash
user    8839  0.0  0.1   2644   932 pts/3    R+   23:01   0:00 ps aux
```

Вывод этой команды ориентирован на пользователя (ключ *u*), но отсортирован по времени запуска процесса. Здесь можно увидеть следующие параметры процесса: пользователь, идентификатор, уровень использования процессора, уровень использования памяти, объём используемой виртуальной памяти, объём используемой реальной памяти, терминал, с которым связан процесс, состояние выполнения, время старта, время исполнения (на процессоре), имя программы и аргументы запуска. Отметим, что сам процесс **ps** всегда находится в конце таблицы, так как сам выполнялся в момент сбора состояния процессов.
При добавлении параметра *-H* можно посмотреть процессы, выстроенные в иерархию:

```
user@desktop ~ $ ps u -H
USER   PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user 22537  0.0  0.3   3720  1560 pts/3    Ss   22:37   0:00 /bin/bash
user  8890  0.0  0.1   2644   876 pts/3    R+   23:13   0:00   ps u -H
user  9243  0.0  0.2   3724  1368 pts/1    Ss+  17:15   0:00 /bin/bash --noediting -i
user  8033  0.0  0.3   3720  1568 tty1     S    13:17   0:00 -bash
user  8168  0.0  0.2   3260  1120 tty1     S+   13:18   0:00   /bin/sh /usr/bin/startx
user  8179  0.0  0.1   2240   636 tty1     S+   13:18   0:00     xinit .xinitrc --
user  8186  0.0  0.2   3516  1080 tty1     S    13:18   0:00       sh
/home/user/.xinitrc
...
```

В данном случае выводятся все процессы текущего пользователя.

pstree

Для иерархического отображения запущеных процессов больше подходит программа **pstree**, отображающая дерево запущенных процессов:

```
user@desktop ~ $ pstree
init─┬─acpid
     ├─5*[agetty]
     ├─2*[artsd]
     ├─cpufreqd
     ├─dcopserver
     ├─events/0
     ├─fcron
     ├─gconfd-2
     ├─hcid
     ├─kaccess
     ├─kded───kded───kded
     ├─kdeinit─┬─artsd───artsd───artsd
     │         ├─kio_file
     │         ├─kio_imap4───kio_imap4
...
```

Видно, что в вершине дерева находится специальный процесс *init*, который в UNIX является самым первым процессом, запускаемым операционной системой.
Эта команда также имеет множество ключей, настраивающих формат вывода информации о процессах.

top

Для вывода динамически изменяющейся информации о процессах и используемых ресурсах системы используется программа **top**. После запуска программы пользователь попадает в интерактивный интерфейс:

```
Tasks:  86 total,   2 running,  84 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.2% us,  0.8% sy,  0.0% ni, 92.0% id,  0.6% wa,  0.3% hi,  0.0% si
Mem:    512480k total,   428352k used,    84128k free,    44848k buffers
Swap:   529160k total,    12312k used,   516848k free,   210444k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8239 dralex    15   0 26856 8860 6512 S  7.9  1.7   5:39.31 artsd
    1 root      16   0  1432  480  420 S  0.0  0.1   0:01.22 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
    3 root      10  -5     0    0    0 S  0.0  0.0   0:00.54 events/0
    4 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
    8 root      10  -5     0    0    0 S  0.0  0.0   0:00.59 kacpid
    7 root       7 -10     0    0    0 S  0.0  0.0   0:00.10 vesafb
...
```

Для выхода используется клавиша **q**. Программа имеет много управляющих клавиш, которые можно узнать, нажав на **h** (help).
Вверху экрана расположена общая информация о системе: число запущенных процессов, уровень использования процессора и памяти. Далее следует обновляемый список процессов в стиле **ps**, отсортированный по уменьшению процессорной активности.

free

Небольшая команда, предоставляющая информацию об использовании памяти. Она представлена не во всех современных версиях UNIX.

```
user@desktop ~ $ free
             total       used       free     shared    buffers     cached
Mem:        512480     476256      36224          0      51216     251712
-/+ buffers/cache:     173328     339152
Swap:       529160      11500     517660
```

Команда выводит объём оперативной памяти (в килобайтах) и устройства своппинга: общий, свободный, занятый, используемый для системных нужд.

df

Существует команда и для простотра заполненности дисков.
Команда **df** выводит сведения обо всех примонтированных файловых системах:

```
user@desktop ~ $ df
Файловая система     1K-блоков      Исп  Доступно  Исп% смонтирована на
/dev/hda6             24697672  21164336   3533336  86% /
udev                    256240       180    256060   1% /dev
/dev/hda1              9827968   8073084   1754884  83% /mnt/win
none                    256240         0    256240   0% /dev/shm
server:/home/shared    8194752   5368992   2416032  69% /mnt/shared
```

Для более «человеческого» представления объёма диска используется ключ `-h`:

```
user@desktop ~ $ df -h
```

```
Файловая система      Разм  Исп  Дост  Исп% смонтирована на
/dev/hda6              24G   21G  3,4G  86% /
udev                  251M  180K  251M   1% /dev
/dev/hda1             9,4G  7,7G  1,7G  83% /mnt/win
none                  251M     0  251M   0% /dev/shm
server:/home/shared   7,9G  5,2G  2,4G  69% /mnt/shared
```

who

Команда выводит список пользователей, работающих в настоящий момент в системе. При запуске без параметров для каждого сеанса пользователя выводится терминал и время входа в систему:

```
user@desktop ~ $ who
user   tty1         2005-10-24 13:17
user   pts/0        2005-10-24 13:19
user   pts/2        2005-10-24 23:59
```

При указании параметра `-u` выводится время неактивности для каждого терминала и идентификатор соответствующего ему процесса:

```
user@desktop ~ $ who
user   tty1         2005-10-24 13:17 11:41        7992
user   pts/0        2005-10-24 13:19 11:41        8228
user   pts/2        2005-10-24 23:59   .          9280
```

mount

Эта команда используется для *монтирования* новых файловых систем. В общем случае команда **mount** имеет следующий формат:

```
mount [-t тип_ФС] имя_устройства точка_монтирования [опции]
```

Рассмотрим пример монтирования гибкого диска:

```
user@desktop ~ $ mount -t vfat /dev/fd0 /mnt/floppy -o rw
user@desktop ~ $ ls /mnt/floppy
document.tex files/
```

В качестве файловой системы указан FAT, опции содержат флаг разрешения чтения и записи содержимого диска. После монтирования файлы доступны в директории `/mnt/floppy`.

Для размонтирования применяется команда **umount**.

При запуске **mount** без параметров выводится список всех примонтированных локальных и сетевых файловых систем:

```
user@desktop ~ $ mount
/dev/hda6 on / type reiserfs (rw,noatime)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
udev on /dev type tmpfs (rw,nosuid)
devpts on /dev/pts type devpts (rw)
/dev/hda1 on /mnt/win type ntfs (ro,uid=1000,gid=100,nls=utf8)
none on /dev/shm type tmpfs (rw)
usbfs on /proc/bus/usb type usbfs (rw,devmode=0664,devgid=85)
server:/home/shared on /mnt/shared type nfs (rw,rsize=32768,wsize=32768,intr,...
```

Можно увидеть ряд служебных файловых систем (proc, udev, sysfs, т.п.).

uname

Вывод имени запущенной UNIX-системы.

```
user@desktop ~ $ uname
Linux
```

Для вывода полного имени используется параметр *-a*:

```
user@desktop ~ $ uname -a
Linux thinkpad 2.6.13-gentoo-r4 #1 Mon Oct 17 12:09:36 MSD 2005 i686
Intel(R)...
```

Команды по работе с процессами

kill

Команда используется для отправки сигнала процессу.

Команда **kill** имеет один из следующих форматов:

```
kill [-s название_сигнала] идентификатор_процесса
kill -название_сигнала идентификатор_процесса
kill -код_сигнала идентификатор_процесса
```

Для того, чтобы отправить сигнал, необходимо знать идентификатор процесса-получателя. Узнать идентификатор можно с помощью команды **ps**. Если при вызове команды **kill** сигнал не указан, то посылается сигнал *TERM*. Обычный пользователь может посылать сигналы только своим процессам. Посылать сигналы процессам других пользователей может только суперпользователь. Рассмотрим пример уничтожения процесса:

```
user@desktop ~ $ ps
PID  TT  STAT      TIME COMMAND
3800  p3  S      0:00,03 su (bash)
3822  p3  T      0:00,08 mplayer /home/guest/music/U96/Das\ Boot.mp3
3824  p3  R+     0:00,00 ps
user@desktop ~ $ kill 3822
```

После вызова команды **kill 3822** процесс *mplayer* будет уничтожен.

Для вывода списка всех сигналов используется команда **kill -l**:

```
user@desktop ~ $ kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
9) SIGKILL     10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGURG
17) SIGSTOP     18) SIGTSTP     19) SIGCONT     20) SIGCHLD
21) SIGTTIN     22) SIGTTOU     23) SIGIO       24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGINFO     30) SIGUSR1     31) SIGUSR2
```

exec

Является встроенной командой оболочки. Используется для замены процесса shell другим процессом.

Команда **exec** имеет следующий синтаксис:

```
exec -lc -a имя исполняемый файл [перенаправление ...]
```

fork (&)

Символ «&» используется для запуска программ в фоновом режиме.

Пример:

```
user@desktop ~ $ mplayer /home/guest/music/U96/Das\ Boot.mp3&
[1] 1558
```

bg, fg u jobs

Являются встроенными командами оболочки. Команды используются для работы с заданиями – процессами, запущенными из командного интерпретатора.
Команда **jobs** имеет следующий синтаксис:

```
jobs [-lnprs] [jobspec ...]
jobs -x команда [аргументы]
```

Для вывода всех заданий используется команда **jobs**:

```
user@desktop ~ $ jobs
[1]+  Running                 nice -n 20 tar -cyf test.tar.bz2 /home/guest/data/* &
[2]+  Stopped                 nice -n 20 tar -cyf test.tar.bz2 /home/guest/texts/*
[3]-  Exit 127                nice 20 tar -cyf /abcdefg/texts ./test.tar.bz2
[4]+  Done                    tar -cyf test.tar.bz2 text.txt
```

Напротив идентификатора задания указано состояние процесса.
Для вывода, кроме основной информации, идентификаторов запущенных процессов используется флаг *-l*. Для вывода только идентификаторов запущенных процессов используется флаг *-p*. Для вывода информации о процессах, у которых изменилось состояние с момента последнего вызова команды **jobs** используется флаг *-n*.
Команда **bg** используется для перевода задачи в фоновый режим.
Команда имеет следующий синтаксис:

```
bg [идентификатор_задачи]
```

Запуск команды без параметров приводит к запуску в фоновом режиме текущей задачи. Если указан параметр *идентификатор_задачи*, то в фоновом режиме будет запущена указанная задача.
Пример:

```
user@desktop ~ $ bg
[1]+ nice -n 20 tar -cyf test.tar.bz2 /home/guest/texts/* &
```

Команда **fg** используется для перевода задачи на передний план.
Команда имеет следующий синтаксис:

```
fg [идентификатор_задачи]
```

Запуск команды без параметров переводит к переводу текущей задачи на передний план. Если указан параметр *идентификатор_задачи*, то задача с указанным идентификатором станет текущей и будет переведена на передний план.
Пример:

```
user@desktop ~ $fg
nice -n 20 tar -cyf test.tar.bz2 /home/guest/texts/*
```

nohup

Команда позволяет запустить процесс, отключив его от управляющего терминала.
Команда **nohup** имеет следующий синтаксис:

```
nohup [--] запускаемая_программа [аргументы]
```

nohup запускает указанную программу с игнорированием сигналов *HUP* и увеличением приоритета для планировщика задач на 5. Таким образом, команда может продолжать выполняться в фоновом режиме и после того, как пользователь выйдет из системы. Если стандартным выводом *stdout* команды является терминал, то он и стандартный поток ошибок *stderr* перенаправляются в файл `nohup.out` в текущей директории. Если это невозможно, то перенаправление происходит в файл `$HOME/nohup.out`. Если и это невозможно сделать, то команда не запускается. Команда **nohup** не переводит автоматически запускаемый процесс в фоновый режим. Чтобы это сделать, необходимо завершить команду символом &.
Пример использования:

```
user@desktop ~ $ nohup mplayer /home/guest/music/U96/Das\ Boot.mp3
appending output to /home/guest/nohup.out
```

nice

Команда используется для запуска процесса с измененным приоритетом для планировщика задач.
Команда **nice** имеет следующий формат:

```
nice [-n смещение_приоритета] запускаемая_программа [аргументы]
```

Команда **nice** позволяет изменять приоритет от -20 (наивысший) до 20 (самый низкий) от текущего. По умолчанию, процессы имеют приоритет командного интерпретатора, из которого они запускаются. Повышать приоритет может только суперпользователь (т.е. задавать отрицательное значение параметра *`-n смещение_приоритета`*). Пример использования команды:

```
user@desktop ~ $ nice -n 15 mplayer /home/guest/music/U96/Das\ Boot.mp3 &
[1] 895
uest@premudrij-peskar$ ps -l
UID   PID  PPID CPU PRI NI   VSZ   RSS MWCHAN STAT  TT       TIME COMMAND
1002   890   889   0   8  0  3112  1660 wait   S     p2    0:00,02 su (bash)
1002   895   890   0 111 15 23000 13200 -      TN    p2    0:00,08 mplayer
/home/guest/music/U96/Das\ Boot.mp3
1002   896   890   0  96  0  1392   768 -      R+    p2    0:00,00 ps -l
```

Значение *NI* процесса *mplayer* равно 15, для всех остальных процессов равно 0 по умолчанию.

renice

Команда используется для изменения приоритета запущенных процессов.
Команда **renice** имеет следующий формат:

```
renice новое_значение_приоритета список_идентификаторов
renice -n смещение_приоритета список_идентификаторов
```

Для изменения значения приоритета отдельных процессов достаточно перечислить их идентификаторы:

```
user@desktop ~ $ ps -l
UID   PID  PPID CPU PRI NI   VSZ   RSS MWCHAN STAT  TT       TIME COMMAND
1002   890   889   0  96  0  3112  1600 -      R     p2    0:00,06 su (bash)
1002   895   890   0 111 17 23000 11224 -      TN    p2    0:00,08 mplayer Boot.mp3
```

```
1002   900   890   0 101  5 23000 11224 -      TN    p2    0:00,08 mplayer Boot.mp3
1002  1084   890   1  96  0  1392   760 -      R+    p2    0:00,00 ps -l

user@desktop ~ $ renice +18 895 900
895: old priority 17, new priority 18
900: old priority 5, new priority 18

user@desktop ~ $ ps -l
UID   PID  PPID CPU PRI NI   VSZ   RSS MWCHAN STAT  TT       TIME COMMAND
1002   890   889   0   8  0  3112  1600 wait   S     p2    0:00,06 su (bash)
1002   895   890   0 111 18 23000 11224 -      TN    p2    0:00,08 mplayer Boot.mp3
1002   900   890   0 101 18 23000 11224 -      TN    p2    0:00,08 mplayer Boot.mp3
1002  1086   890   0  96  0  1392   760 -      R+    p2    0:00,00 ps -l
```

Для изменения приоритета всех процессов какого-либо пользователя необходимо указать флаг *-u*. В этом случае значения идентификаторов после флага *-u* будут интерпретироваться как идентификаторы пользователей. Можно задавать как числовые, так и символьные идентификаторы пользователей.

```
user@desktop ~ $ renice -n 1 -u guest
1002: old priority 0, new priority 1
user@desktop ~ $ ps -l

UID   PID  PPID CPU PRI NI   VSZ   RSS MWCHAN STAT  TT       TIME COMMAND
1002   890   889   0   8  1  3112  1600 wait   SN    p2    0:00,07 su (bash)
1002   895   890   0 111 18 23000 11224 -      TN    p2    0:00,08 mplayer Boot.mp3
1002   900   890   0 101 18 23000 11224 -      TN    p2    0:00,08 mplayer Boot.mp3
1002  1110   890   0  97  1  1392   760 -      RN+   p2    0:00,00 ps -l
```

Из данного примера видно, что значения *NI* для процессов *mplayer* не изменились. Вызов команды **renice** с флагом *-n* изменяет только минимальные приоритеты (в данном случае, нули).