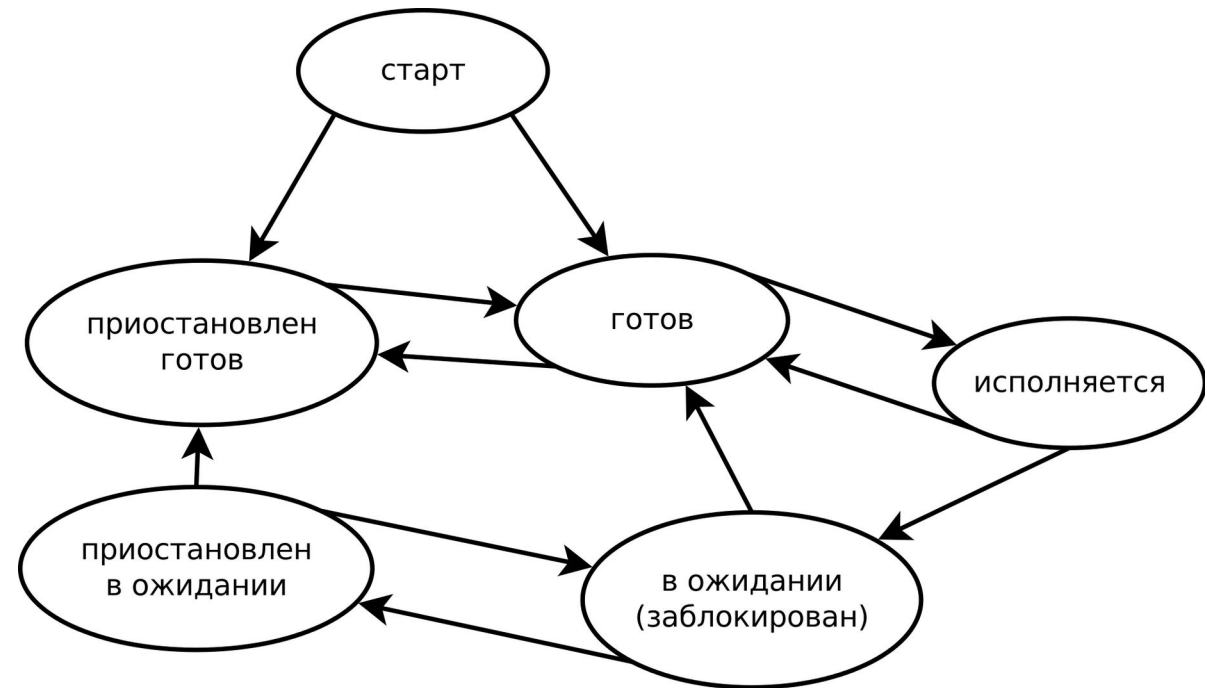


Процессы в Linux



Понятие процесса

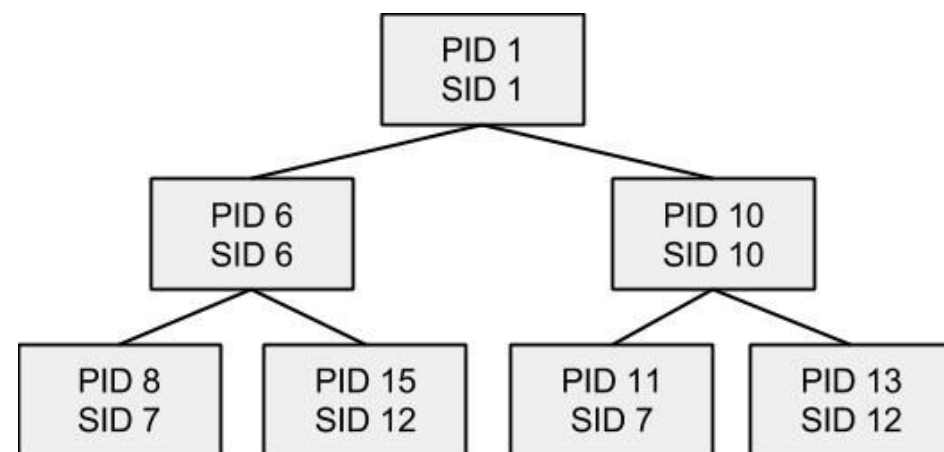
Сущность процесса неразрывно связана с многозадачностью операционной системы. Если запускаемая программа будет отбирать все процессорное время системы, то коэффициент полезного использования ЦП, и УВВ будет приближен к нулю. Решение проблемы привело к созданию многозадачных операционных систем



Зачем нужен параллелизм на уровне процессов?

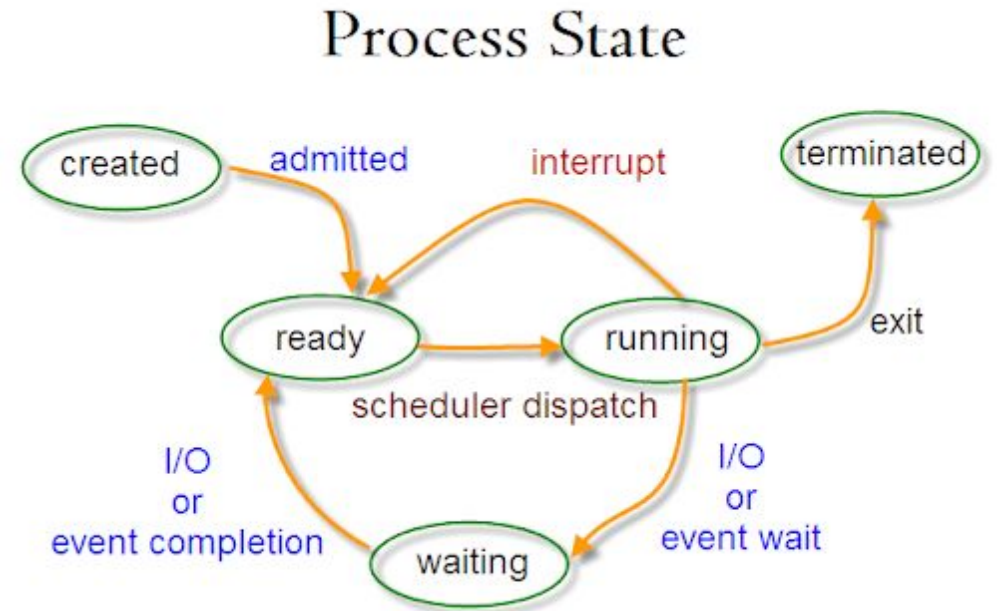
Дерево процессов

При загрузке ядро выполняет всего одну программу —обычно это `/sbin/init` (`systemd`). Процесс `init` отвечает за запуск всех основных задач в Linux, таких как управление входом в систему и постоянно работающие программы-демоны. Потомок программы `init`, в свою очередь, может запускать собственные дочерние программы. Результатом является древовидная иерархия процессов



Состояние процесса

- Рождение процесса
- Состояние «готов»
- Состояние «выполняется»
- Перерождение в другую программу
- Состояние «ожидает»
- Состояние «остановлен»
- Завершение процесса
- Состояние «зомби»
- Забытье



Приоритетные и фоновые процесс

Обычно терминалы Linux посылают сигнал SIGINT текущему процессу переднего плана при нажатии комбинации клавиш CTRL-C.

Все новые задания запускаются в активном режиме

Запуск в фоновом режиме завершать командную строку символом &

Для этого существует сигнал SIGTSTP. Его можно отправить при помощи комбинации клавиш CTRL-Z

fg bg jobs

Как запустить процесс в фоне?

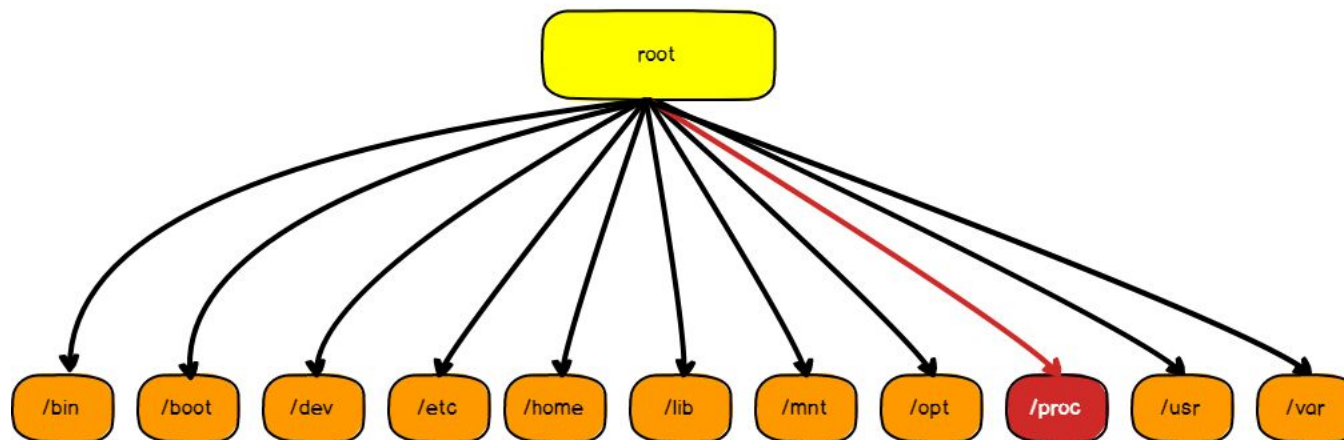
Как сделать текущий активный процесс фоновым?

procfs

Специальная файловая система, используемая в UNIX-подобных операционных системах. Позволяет получить доступ к информации из ядра о процессах.

proc - это совсем необычная директория, не существует на диске или даже в оперативной памяти, файлы и хранящаяся в них информация генерируется ядром на лету

Linux Filesystem



Структура файловой системы procfs

/proc/buddyinfo - информация о фрагментации памяти

/proc/cgroups - система контейнеризации

/proc/cmdline - параметры запуска ядра загрузчиком

Grub

/proc/config.gz - текущая конфигурация ядра

/proc/consoles - зарегистрированные в ядре

символические устройства

/proc/cpuinfo - информация о процессоре

94	cmdline	keys	scsi
945	config.gz	key-users	self
949	consoles	kmsg	slabinfo
95	cpuinfo	kpagecgroup	softirqs
96	crypto	kpagecount	stat
965	devices	kpageflags	swaps
97	diskstats	latency_stats	sys
970	dma	loadavg	sysrq-trigger
973	driver	locks	sysvipc
98	dynamic_debug	meminfo	thread-self
987	execdomains	misc	timer_list
99	fb	modules	tty
994	filesystems	mounts	uptime
998	fs	mtrr	version
acpi	interrupts	net	vmallocinfo
asound	iomem	pagetypeinfo	vmstat
bootconfig	ioports	partitions	zoneinfo
buddyinfo	irq	pressure	
bus	kallsyms	sched_debug	
cgroups	kcore	schedstat	

Структура файловой системы procfs

/proc/crypto - криптографические шифры

/proc/devices - блочные и символические устройства
подключенные к системе

/proc/diskstats - Статистика ввода и вывода на блочные
устройства

/proc/fb- устройства фреймбуфера

/proc/ consoles - зарегистрированные в ядре
символические устройства

/proc/filesystems - список файловых систем

```
94      cmdline      keys      scsi
945     config.gz     key-users self
949     consoles     kmsg      slabinfo
95      cpuinfo      kpagecgroup softirqs
96      crypto       kpagecount stat
965     devices      kpageflags swaps
97      diskstats    latency_stats sys
970     dma          loadavg    sysrq-trigger
973     driver        locks      sysvipc
98      dynamic_debug meminfo    thread-self
987     execdomains  misc       timer_list
99      fb           modules    tty
994     filesystems  mounts     uptime
998     fs           mtrr       version
acpi    interrupts     net        vmallocinfo
asound  iomem          pagetypeinfo vmstat
bootconfig ioports        partitions zoneinfo
buddyinfo irq            pressure
bus     kallsyms      sched_debug
cgroups kcore          schedstat
```

Структура файловой системы procfs

/proc/interrupts - прерывания

/proc/iomem - карта памяти для всех программ

/proc/ioports - Статистика ввода и вывода на блочные устройства

/proc/kallsyms- функции и их адреса

/proc/kcore -содержимое оперативной памяти

/proc/kmsg - сообщений ядра

/proc/kpagecount - размер одной страницы

/proc/loadavg - среднюю нагрузку на систему

```
94      cmdline      keys      scsi
945     config.gz     key-users self
949     consoles     kmsg      slabinfo
95      cpuinfo      kpagecgrou softirqs
96      crypto      kpagecount stat
965     devices     kpageflags swaps
97      diskstats   latency_stats sys
970     dma          loadavg    sysrq-trigger
973     driver       locks      sysvipc
98      dynamic_debug meminfo    thread-self
987     execdomains  misc      timer_list
99      fb           modules    tty
994     filesystems  mounts     uptime
998     fs           mtrr       version
acpi    interrupts     net        vmallocinfo
asound  iomem           pagetypeinfo vmstat
bootconfig ioports         partitions zoneinfo
buddyinfo irq            pressure
bus     kallsyms      sched_debug
cgroups kcore          schedstat
```

Структура файловой системы procfs

/proc/locks - заблокированных ядром ресурсов

/proc/meminfo - информация об оперативной памяти и пространстве подкачки

/proc/misc - перечислены различные драйверы

/proc/modules - модулей ядра

/proc/mounts - точки монтирования

/proc/partitions - разделы жестких дисков

/proc/stat - статистическая информация

```
94      cmdline      keys      scsi
945     config.gz     key-users self
949     consoles     kmsg      slabinfo
95      cpuinfo      kpagecgrou softirqs
96      crypto      kpagecount stat
965     devices     kpageflags swaps
97      diskstats   latency_stats sys
970     dma          loadavg    sysrq-trigger
973     driver       locks      sysvipc
98      dynamic_debug meminfo    thread-self
987     execdomains  misc      timer_list
99      fb          modules    tty
994     filesystems  mounts     uptime
998     fs           mtrr       version
acpi    interrupts     net        vmallocinfo
asound  iomem          pagetypeinfo vmstat
bootconfig ioports        partitions zoneinfo
buddyinfo irq            pressure
bus     kallsyms      sched_debug
cgroups kcore         schedstat
```

Структура файловой системы procfs

/proc/swaps - информация о пространстве подкачки

/proc/sysrq-trigger - передача ядру специальных SysRq команд

/proc/uptime - перечислены различные драйверы

/proc/version - версию ядра, компилятора

/proc/vmstat - информация о виртуальной памяти

/proc/zoneinfo — как в vmstat, но с разбиением на зоны памяти

94	cmdline	keys	scsi
945	config.gz	key-users	self
949	consoles	kmsg	slabinfo
95	cpuinfo	kpagecgroup	softirqs
96	crypto	kpagecount	stat
965	devices	kpageflags	swaps
97	diskstats	latency_stats	sys
970	dma	loadavg	sysrq-trigger
973	driver	locks	sysvipc
98	dynamic_debug	meminfo	thread-self
987	execdomains	misc	timer_list
99	fb	modules	tty
994	filesystems	mounts	uptime
998	fs	mtrr	version
acpi	interrupts	net	vmallocinfo
asound	iomem	pagetypeinfo	vmstat
bootconfig	ioports	partitions	zoneinfo
buddyinfo	irq	pressure	
bus	kallsyms	sched_debug	
cgroups	kcore	schedstat	

Структура файловой системы procfs

Файловая система proc состоит не только из файлов, но здесь есть и папки. Больше всего здесь папок с номерами вместо имен. Каждый номер - PID процесса

cmdline - содержит команду с помощью которой был запущен процесс, а также переданные ей параметры

cwd - символическая ссылка на текущую рабочую директорию процесса

exe - ссылка на исполняемый файл

root - ссылка на папку суперпользователя

environ - переменные окружения, доступные для процесса

fd - содержит файловые дескрипторы, файлы и устройства, которые использует процесс

maps, statm, и mem - информация о памяти процесса

stat, status - состояние процесса

Как узнать путь до исполняемого файла, зная pid процесса, используя procfs?

Команда ps

Команда `ps` выводит список текущих процессов на вашем сервере

Без аргументов `ps` показывает запущенные процессы, выполняемые пользователем в окне терминала

Чтобы просмотреть все запущенные процессы, используйте одну из следующих команд: `ps -e` или `ps -A`

Все процессы, кроме лидеров сессии: `ps -d`

только процессы, связанные с этим терминалом: `ps T`

Если вы хотите просмотреть все работающие (running) процессы: `ps r`

Если знаете идентификаторы процессов PID: `ps -p <pid1>,...,<pidN>`

Поиск по имени команды: `ps -C <команда>`

Команда ps

все процессы, выполняемые группой: `ps -G <groupname>`

все процессы, выполняемые группой по id: `ps -g <groupid>`

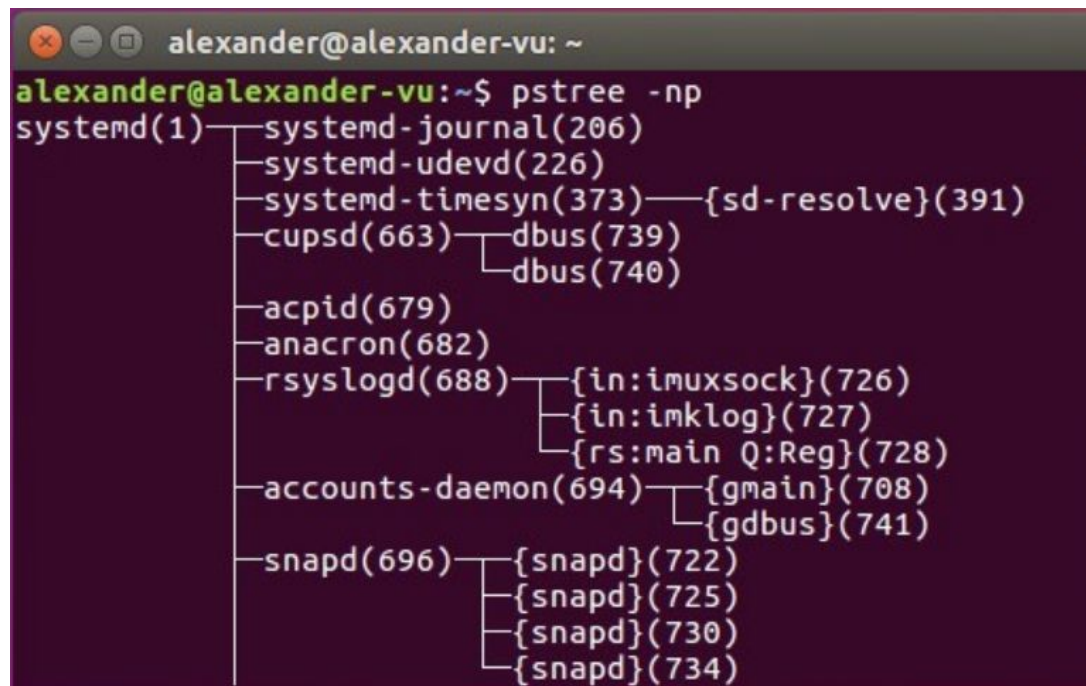
Больше выводимых столбцов: `ps -ef`

Указать формат вывода(список колонок): `ps -e --format <format>`

Указать порядок сортировки колонок `ps -ef --sort <sortcolumns>`

Команда pstree

В простейшей форме, когда вызывается без какой-либо опции или аргумента, pstree отображает иерархическую древовидную структуру всех запущенных процессов:



```
alexander@alexander-vu: ~  
alexander@alexander-vu:~$ pstree -np  
systemd(1)─systemd-journal(206)  
           └─systemd-udev(226)  
             └─systemd-timesyn(373)─{sd-resolve}(391)  
           └─cupsd(663)─dbus(739)  
                   └─dbus(740)  
           └─acpid(679)  
           └─anacron(682)  
           └─rsyslogd(688)─{in:imuxsock}(726)  
                           └─{in:imklog}(727)  
                           └─{rs:main Q:Reg}(728)  
           └─accounts-daemon(694)─{gmain}(708)  
                                   └─{gdbus}(741)  
           └─snapd(696)─{snapd}(722)  
                       └─{snapd}(725)  
                       └─{snapd}(730)  
                       └─{snapd}(734)
```

Команда pstree

pstree объединяет идентичные ветви, используются квадратные скобки

отключить объединение идентичных веток, используйте параметр -c

```
devconnected@devconnected:~$ pstree | head -n 5
systemd-+-ModemManager---2*[{ModemManager}]
        |-NetworkManager---2*[{NetworkManager}]
        |-accounts-daemon---2*[{accounts-daemon}]
        |-acpid
        |-anacron
devconnected@devconnected:~$
```

Процессы, принадлежащие этому пользователю:

```
pstree andreyex
```

В качестве корня дерева процесс с указанным PID:

```
pstree 2245
```

Команда pstree

Показать PID процессов:

```
pstree -p
```

Сортировка по PID:

```
pstree -pn
```

Показать PGID:

```
pstree -g
```

Показать аргументы командной строки:

```
pstree -a
```

Команда pgrep

Позволяет искать процессы соответствующие паттерну поиска:

`pgrep [OPTIONS] <PATTERN>`

Пример: `pgrep ssh`

Использовать другой разделитель: `pgrep ssh -d ' '`

Опция `-l` указывает pgrep показать имя процесса: `pgrep ssh -l`

Пример с регуляркой: `pgrep '^ssh$' -l`

Проверять строку вместе с параметрами: `pgrep -f ssh`

Процессы принадлежащие пользователю: `pgrep -u root`

Процессы, которые не соответствуют заданным критериям `-v` : `pgrep -v -u mark`

Опция `-c` Показать только кол—во процессов удовлетворяющих поиску: `pgrep -c ssh`

Команда top

Команда top в Linux системах позволяет вывести в виде таблицы перечень запущенных процессов

```
viktor@viktor-VirtualBox: ~  
top - 18:00:32 up 1 min, 1 user, load average: 1.61, 0.91, 0.35  
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 1.5 us, 1.5 sy, 0.0 ni, 96.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 5945.8 total, 4258.3 free, 611.1 used, 1076.4 buff/cache  
MiB Swap: 1162.4 total, 1162.4 free, 0.0 used. 5081.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	167736	11588	8308	S	0.0	0.2	0:01.20	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0-events
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblockd
7	root	20	0	0	0	0	I	0.0	0.0	0:00.03	kworker/0:1-events
8	root	20	0	0	0	0	I	0.0	0.0	0:00.01	kworker/u8:0-events_unbound
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
11	root	20	0	0	0	0	I	0.0	0.0	0:00.15	rcu_sched
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
13	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1

Команда htop

htop — продвинутый монитор процессов, написанный для Linux. Он был задуман заменить стандартную программу top

```
 1  [|||||] 16.4% Tasks: 80, 168 thr; 2 running
 2  [|||] 4.0% Load average: 1.17 2.31 2.42
 3  [|] 2.7% Uptime: 1 day, 05:11:34
Mem[|||||] 1007/7479MB
Swp[|] 0/8099MB

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
16499 joe     20   0 1904M  325M  73396 S 13.7  4.4   3:26.71 cinnamon --replace
16157 root    20   0  305M  74336 37616 S  3.9  1.0  16:52.70 /usr/bin/X :0 -audi
18419 joe     20   0  25784  3516  2888 R  1.3  0.0   0:00.35 htop
17369 joe     20   0   581M  27724 21680 S  1.3  0.4   0:13.16 gnome-terminal
17391 joe     20   0  1363M  291M  47740 S  0.7  3.9   1h11:10 simplescreenrecorde
18077 joe     20   0  1363M  291M  47740 S  0.7  3.9   2:12.44 simplescreenrecorde
18079 joe     20   0  1363M  291M  47740 S  0.7  3.9   0:02.26 simplescreenrecorde
18421 joe     20   0   443M  22788 18848 S  0.0  0.3   0:00.25 gnome-screenshot --
18080 joe     20   0  1363M  291M  47740 S  0.0  3.9   0:05.40 simplescreenrecorde
16501 joe     20   0  1904M  325M  73396 S  0.0  4.4   0:07.82 cinnamon --replace
18078 joe     20   0  1363M  291M  47740 S  0.0  3.9   0:39.42 simplescreenrecorde
16531 joe     20   0  1904M  325M  73396 S  0.0  4.4   0:00.09 cinnamon --replace
16439 joe     20   0   886M  40912 32292 S  0.0  0.5   0:01.20 /usr/lib/x86_64-lin
  1 root    20   0  33884  4244  2552 S  0.0  0.1   0:01.79 /sbin/init
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```


Команда htop

htop — продвинутый монитор процессов, написанный для Linux. Он был задуман заменить стандартную программу top

```
 1  [|||||] 16.4% Tasks: 80, 168 thr; 2 running
 2  [||] 4.0% Load average: 1.17 2.31 2.42
 3  [|] 2.7% Uptime: 1 day, 05:11:34
Mem[|||||1007/7479MB]
Swp[0/8099MB]

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
16499 joe     20   0 1904M  325M  73396 S 13.7  4.4   3:26.71 cinnamon --replace
16157 root    20   0  305M  74336 37616 S  3.9  1.0  16:52.70 /usr/bin/X :0 -audi
18419 joe     20   0  25784  3516  2888 R  1.3  0.0   0:00.35 htop
17369 joe     20   0  581M  27724 21680 S  1.3  0.4   0:13.16 gnome-terminal
17391 joe     20   0 1363M  291M  47740 S  0.7  3.9  1h11:10 simplescreenrecorde
18077 joe     20   0 1363M  291M  47740 S  0.7  3.9  2:12.44 simplescreenrecorde
18079 joe     20   0 1363M  291M  47740 S  0.7  3.9  0:02.26 simplescreenrecorde
18421 joe     20   0  443M  22788 18848 S  0.0  0.3   0:00.25 gnome-screenshot --
18080 joe     20   0 1363M  291M  47740 S  0.0  3.9  0:05.40 simplescreenrecorde
16501 joe     20   0 1904M  325M  73396 S  0.0  4.4  0:07.82 cinnamon --replace
18078 joe     20   0 1363M  291M  47740 S  0.0  3.9  0:39.42 simplescreenrecorde
16531 joe     20   0 1904M  325M  73396 S  0.0  4.4  0:00.09 cinnamon --replace
16439 joe     20   0  886M  40912 32292 S  0.0  0.5   0:01.20 /usr/lib/x86_64-lin
  1 root    20   0  33884  4244  2552 S  0.0  0.1   0:01.79 /sbin/init
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```


Приоритеты процессов

Переключение центрального процессора выполняет специальная компонента подсистемы управления процессами, называемая планировщиком (scheduler)

По умолчанию для пользовательских задач используется вытесняющий алгоритм CFS (completely fair scheduler). Для каждой задачи определяется выделяемая справедливая (в соответствии с ее относительным «приоритетом») доля процессорного времени

Для дифференциации задач используют 40 относительных POSIX-приоритетов на шкале от —20 (наивысшим) до +19(низшим), называемых «любезностью» задачи NICE

Приоритеты процессов

Переключение центрального процессора выполняет специальная компонента подсистемы управления процессами, называемая планировщиком (scheduler)

По умолчанию для пользовательских задач используется вытесняющий алгоритм CFS (completely fair scheduler). Для каждой задачи определяется выделяемая справедливая (в соответствии с ее относительным «приоритетом») доля процессорного времени

Для дифференциации задач используют 40 относительных POSIX-приоритетов на шкале от —20 (наивысшим) до +19(низшим), называемых «любезностью» задачи NICE

Команда nice.

Утилита nice — программа в UNIX-подобных ОС, предназначенная для запуска процессов с изменённым приоритетом nice

`nice [-n смещение] [--adjustment=смещение] [команда [аргумент...]]`

Примерчик с `bubble_sort.py`

Алгоритмы планировщика Linux*

Кроме приоритетной очереди, планировщик Linux позволяет использовать еще три алгоритма планирования — FIFO, RR и EDF, предназначенные для задач реального времени.

Вытесняющий алгоритм RR (round robin) организует простейшее циклическое обслуживание с фиксированными квантами времени

FIFO (first in first out) является его невытесняющей модификацией RR

Алгоритм EDF (Earliest Deadline First) предназначен для обеспечения гарантий периодическим задачам реального времени, которым важно получать периодическое обслуживание так, чтобы задача не была вытеснена в течение определенного времени.

Сигналы и команда kill

Сигналы - один из способов межпроцессного взаимодействия в Unix

Наиболее типичный синтаксис команды kill имеет следующий вид:

kill [-сигнал] PID...

Номер	Название	Описание
1	HUP	Обрыв связи
2	INT	Прервать
9	KILL	Уничтожить
15	TERM	Завершить
18	CONT	Продолжить
19	STOP	Приостановить
3	QUIT	Выйти
11	SEGV	Ошибка сегментации
20	TSTP	«Стоп» с клавиатуры
28	WINCH	Изменение окна

Команда killall

Кроме того, существует возможность с помощью команды killall послать сигнал сразу нескольким процессам

killall [-и пользователь] [-сигнал] имя..

Помните: так же как при использовании команды kill, вы должны обладать привилегиями суперпользователя, чтобы посылать сигналы процессам, которыми невладеете.

Спасибо за внимание!

