Университет ИТМО, кафедра ВТ

Лабораторная работа №3 по “Системному Программному Обеспечению”

Работу выполнил
студент группы P3200
Рогов Я. С.

Преподаватель
Дергачёв А.М.

Санкт-Петербург, 2016

Задание 1: пример запуска команды 'vi /etc/passwd' в фоновом режиме

Перевод программы в фоновый режим при запуске выполняется с помощью оператора &.

```
vi /etc/passwd &
```

Задание 2: пример перевода команды 'vi /etc/passwd' в фоновый режим

Запустим vi с помощью команды

```
vi /etc/passwd
```

Для перевода vi в фоновый режим, необходимо нажать сочетания клавиш Ctrl-Z в режиме ввода команд. Выведется сообщение:

```
[1] + Stopped (SIGTSTP)        vi
```

Для возвращения в программу необходимо воспользоваться утилитой **fg**.

Задание 3: описание изученных команд с основными ключами и примерами их использования с пояснением

ps – выводит информацию об активных процессах. По-умолчанию (без к лючей) выводит список процессов, принадлежащих вызывающему команду пользователя и запущенных из данного терминала.

Основные ключи:

-a – выводит все процессы, что не являются лидерами сессии (session leader) и не ассоциированы с данным терминалом

-l – выводит подробную таблицу данных. Столбцы таблицы:

F – (устар.) флаги

S – состояние процесса: 0 (выполняется), S (спящий режим), R (в очереди на выполнение), T (остановлен), W (ожидает), Z (состояние "зомби": процесс завершён, однако родительский процесс ещё не принял значение кода выхода)

UID – ID пользователя, вызвавшего процесс

PID – ID процесса

PPID – ID родительского процесса

C – (устар.) использование процессора

PRI – приоритет процесса

NI – значение "nice", используемое для планировки по приоритетам.

ADDR – адрес процесса в памяти

SZ – количество памяти, занимаемое процессом

WCHAN – адрес события, которого ждёт спящий процесс

TTY – управляющий терминал для данного процесса

TIME – время работы процесса

CMD – команда, включающая в себя имя и переданные аргументы

-**A,-e –** выводит список всех процессов. При указании этих ключей, ниже описанные ключи игнорируются.

Фильтры процессов по ID/др.:

-g <u>списокPID</u> – PID лидера группы процесса есть в списке

-G <u>списокGID</u> – настоящий GID вызвавшего процесс есть в списке

-t <u>терминал</u> – процесс ассоциирован с <u>терминалом</u>

-u <u>списокUID</u> – эффективный UID вызвавшего процесс есть в списке

-U <u>списокUID</u> – реальный UID вызвавшего процесс есть в списке

-p <u>списокPID</u> – ID процесса есть в списке

-s <u>списокPID</u> – PID лидера сессии есть в списке

Примеры:

`ps -l` – выводит все процессы системы

`ps -lukorg -Gstuds` – выводит процессы, запущенные пользователем korg или пользователями группы studs

crontab (и демон cron) – управление таблицами планировщика (crontab-файл) задач и его демон соответственно. (Демон – программа, запускаемая как фоновый процесс и выполняющая некоторую функциональность. Управление происходит с помощью специальных утилит или файлов конфигурации, ассоциированных с демоном).

Основные ключи:

-**e –** создаёт копию crontab пользователя, или создаёт пустой файл при его отсутствии, для редактирования. По завершению редактирования устанавливает файл как crontab-файл пользователя

-l – выводит содержимое crontab-файла пользователя.

-r – удаляет crontab пользователя

Формат записей в файлах crontab:

`ММ ЧЧ ДД ММ ДН Команда – минуты, часы, дни, месяцы, дни недели, и команда для исполнения. В каждом из первых пяти полей можно указать:`

`число – выполнять команду в определённое время`

`число1,число2,... - выполнять команду в каждое определённое время, указанное в списке`

`число1-число2 – выполнять команду в каждое определённое время, попадающее в данный диапазон значений`

`*(звёздочка) – выполнять команду в каждое определённое время, заданное границами для данного поля.`

`Например, шаблон "1 3,4 10-12 * *" соответствует выполнению команды каждое 10,11,12 числа месяца в 3:01 и 4:01`

Примеры использования crontab:

`crontab -e someuser – отредактировать crontab-файл пользователя someuser (доступно только для root-пользователя и администраторов системы)`

Примеры использования crontab-файлов (пример записей в них):

`0 22 * * * write Dave <<< 'Time to sleep. Good Night.' && shutdown` – каждый день в 22:00 сообщение отсылается пользователю Dave и система завершает работу.

`0 0 1 1 *  wall <<< "Happy New  Year, Dear Friends!"` - поздравляет с Новым Годом всех пользователей системы.

at – позволяет задать (через стандартный ввод) команды для отложенного (по времени) выполнения.

Основные ключи:

-c/-k/-s – устанавливает для исполнения команды csh/ksh/sh соответственно

-f файл – задаёт файл с командами для выполнения

-l/-l at_ID – выводит все отложенные команды/команды с данным ID пользователя

-r at_ID – удалить отложенную команду с данным ID

-t время – время выполнения команды в формате ГГГГ-ММ-ДД чч:мм:сс

Примеры:

`at -k now + 2 hours <<< 'write Dave <<< "take a rest"'` – написать сообщение пользователю через 2 часа, используя ksh для исполнения командыq

nice – запустить команду с другим приоритетом планировки.

Основные ключи:

–<u>число</u>, -n <u>число</u> – установить изменение приоритета на данное целое число. По-умолчанию (если не указывать число) оно равно 10. Отрицательное число может передать только суперпользователь (иначе игнорируется)

Примеры:

`nice -19 tar xf archive.tar` – выполнить распаковку архива с минимальным приоритетом (т.о. минимально мешая работе основных процессов)

nohup – создаёт с переданными аргументами/меняет существующий процесс, чтобы он игнорировал сигнал SIGHUP.

Основные ключи:

-g <u>GPID</u> – делает группы процессов устойчивыми к сигналу

-p <u>PID</u> – делает запущенные процессы устойчивыми к сигналу

-**F –** меняет устойчивость, даже если есть контроль со стороны другого процесса

Примеры:

`nohup -p $(pgrep -P $$) -` все процессы, созданные из текущего терминала, будут игнорировать сигнал SIGHUP

kill – посылает определённый сигнал процессам. По-умолчанию – SIGTERM. Только для процессов того же пользователя (если не суперпользователь).

Основные ключи:

-l / -l <u>кодвыхода</u>– выводит список всех поддерживаемых утилитой сигналов/выводит название сигнала, из-за которого программа закончилас с данным кодом.

-<u>имясигнала</u>, -s <u>имясигнала</u>/-<u>номерсигнала</u> – позволяет указать сигнал через его название/номер.

Примеры:

```
kill -kill $(ps -u$(whoami) -o pid=)
```

fg - перевести процесс из фонового режима и продолжить его выполнение.

Основные ключи: -

Использование:

Вызов команды имеет вид 'fg [job_id]' или 'fg [%expr]' , где job_id – PID процесса. %expr – одно из выражений:

%<u>число</u> – аналогично job_id

%<u>строка</u> – любой процесс, чья команда начинается со <u>строки</u>.

%% - текущее задание

%+ - аналогично %%

%- - предыдущее задание

По-умолчанию - %%.

Примеры:

`fg %vim` – продолжить работу в первом найденном в списке заданий редакторе vim

bg – перевести процесс в фоновый режим и продолжить его выполнение.

Основные ключи: -

Использование: см. **fg**

Примеры:

bg %tr – возобновить работу первое найденное задание с командой 'tr'

jobs – отображает список заданий для данной оболочки.

Основные ключи:

-l – подробный режим отображения: номер задания, настоящее задание, GPID, состояние и саму команду.

-n – показывает только остановленные и завершённые с момента последнего опроса состояния задания.

-p – отображает только PID лидеров групп выбранных процессов.

Примеры:

`jobs -l` – если перед этой командой запустить, например, команду `'tr a b > file &'`, то вывод будет следующим:

```
s207220@helios:/home/s207220$ jobs -l
[2] + 16207     Stopped (SIGTTIN)        tr a b > file &
```

priocntl – управление параметрами планировки у процессов. Возможны 4 варианта использования: просмотр доступных классов планировки (**-l**), просмотр параметров у уже запущенных процессов (**-d**), изменение параметров у уже запущенных процессов (**-s**) и запуск команд с определёнными параметрами (**-e**).

Основные ключи:

-l – отображение доступных классов процессов

-d – отобразить параметры планировки для уже запущенных процессов

-s – измененить параметры у уже запущенных процессов

-e – запуск команды с указанными параметрами

-i <u>типID</u> – позволяет указать <u>тип ID</u>, которые указываются в списке ID (последний аргумент). Для ключей **-d** и **-s**. Возможные типы:

all – применяется ко <u>всем</u> существующим процессам

class – применяется ко всем процессам данного класса (см. ниже)

gid, pgid, pid, ppid, sid – соответствующие типы ID в unix

-c <u>класс</u> – указывает класс для процесса. Для ключей **-s** и **-e**.

Классы процессов – название групп, на которые классифицируются все процессы на основе их нужд. Например, класс SYS описывает системных процессов. Необходимы для удобного разделения процессов по их предназначению и последующему применению разных политик планировки на основе их, собственно, классов. Основные классы:

RT – процессы "реального времени". Имеют наивысший приоритет исполнения, т.к. задачи, выполняемые ими, требуют незамедлительного и определённого (по времени) отклика.

SYS – системные процессы. Имеют второй по значимости приоритет исполнения

TS – процессы с честным разделением процессорного времени (time-sharing)

FX – процессы с фиксированным приоритетом

IA – интерактивные процессы, требующие хорошего времени отклика

FS – процессы с "честным разделением" (fair-share) ресурсов

Задание 4: таблица изученных сигналов с номером сигнала, кратким именем, описанием и стандартной реакцией на сигнал

Сигнал (SIG*)	Номер	Стандартная реакция	Описание
HUP	1	Выход	Остановка управляющего терминала или процесса
INT	2	Выход	Прерывание процесса (комбинация Ctrl-C)
QUIT	3	Выход (ядро)	Выход с сохранением дампа ядра
ILL	4	Выход (ядро)	Неправильная/неизвестная/ недоступная (по правам) инструкция
TRAP	5	Выход (ядро)	Встреча точки останова
ABRT	6	Выход (ядро)	Аварийное завершение процесса
FPE	8	Выход (ядро)	Арифметическая ошибка. Например, деление на 0
KILL	9	Выход (н)	Немедленное завершение процесса
BUS	10	Выход (ядро)	(Аппаратная) Ошибка доступа к памяти
SEGV	11	Выход (ядро)	Ошибка доступа к виртуальной памяти
SYS	12	Выход (ядро)	Неправильный системный вызов (переданы неправильные аргументы)
PIPE	13	Выход	Попытка записи в неподсоединённый с другой стороны (для чтения) именованный канал (пайп)
ALRM	14	Выход	Сигнал-таймер
TERM	15	Выход	Завершение процесса
CHLD	18	Игнорирование	Изменение состояние дочернего процесса
STOP	23	Остановка (н)	Остановка процесса оп. системой
TSTP	24	Остановка	Остановка процесса пользователем (Ctrl-Z)
CONT	25	Игнорирование, Продолжение	Продолжение выполнения процесса, выполняемое ОС. (После сигналов 23/24)
TTIN	26	Остановка	Остановка процесса после попытки чтения из несуществующего потока ввода
TTOU	27	Остановка	Остановка процесса после попытки записи в несуществующий поток вывода

Выход – завершение работы процесса (аналогично сис. вызову exit)

Выход (ядро) – завершение работы процесса (аналогично сис. вызову exit) с созданием дампа ядра в рабочей директории.

Игнорирование – процесс игнорирует сигнал

Остановка – приостановка работы процесса

Продолжение – возобновление работы процесса.

(н) – неблокируемый сигнал. Процесс не может обработать его или игнорировать, обрабатывается на уровне ОС.

Вывод

В ходе данной лабораторной работы я познакомился со средствами планировки заданий в Unix системах (и Solaris в частности), позволяющих более удобное управление процессами в системе, а также я ознакомился с концепцией сигналов и изучил основные сигнал, используемые в Unix системах.