Основы Unix

Часть 2. Процессы. SSH

Процессы в Unix

Процесс - это экземпляр работающей программы.

- Процессами управляет модуль ядра. Включает в себя планировщик выполнения, управление памятью для процесса и отвечает за межпроцессорное взаимодействие.
- Каждый процесс запущен от имени конкретного пользователя
- Каждый процесс думает, что он единственный
- Каждый запущенный процесс имеет уникальный process ID - PID (5-значный идентификатор уникальный для конкретного времени, но потом повторяется)
- Процессы могут запускаться в фоновом (daemon) или интерактивном режиме (ожидается ввод со стороны пользователя)
- Процессы могут порождаться другими процессами

Диаграмма состояний процессов (1)

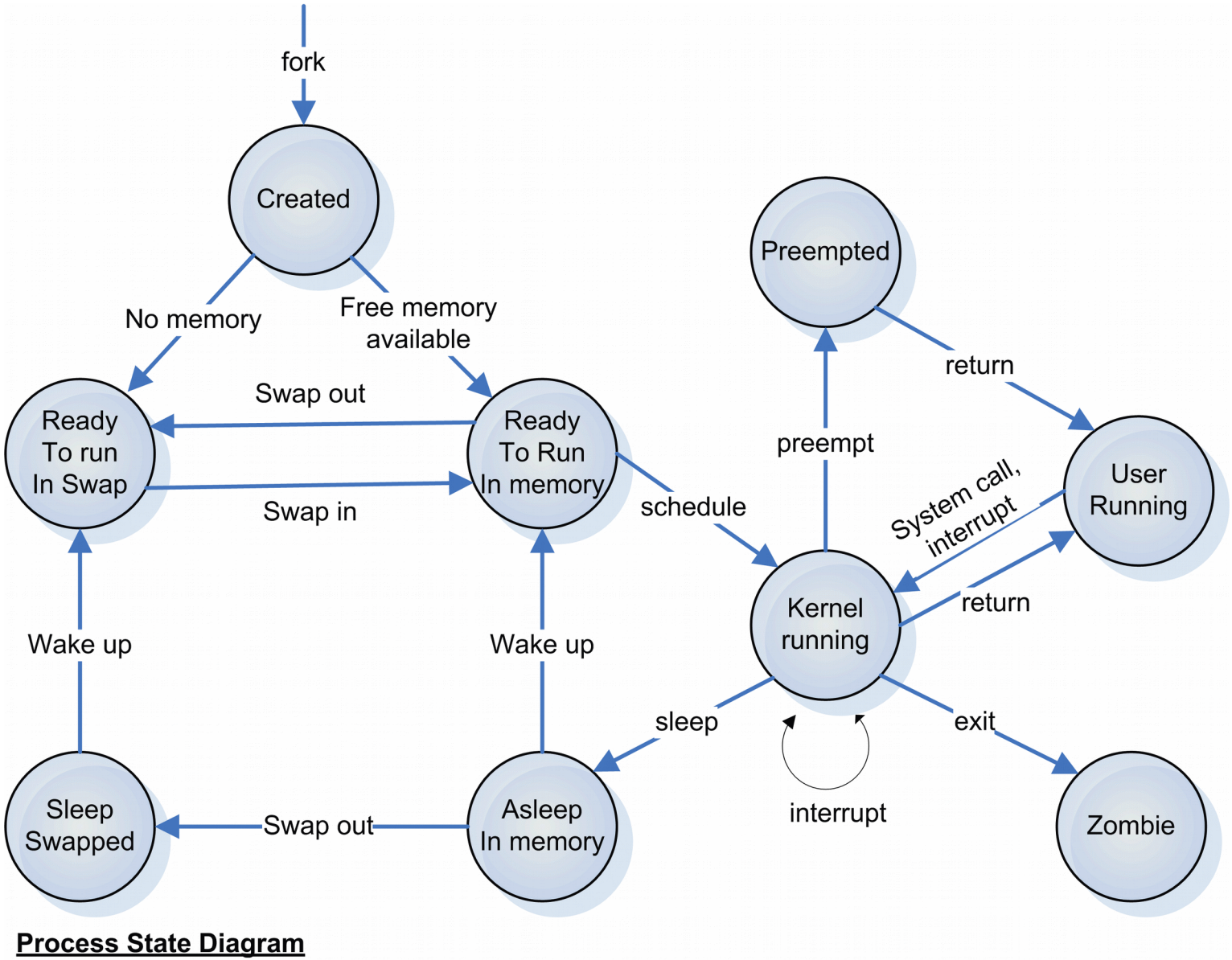
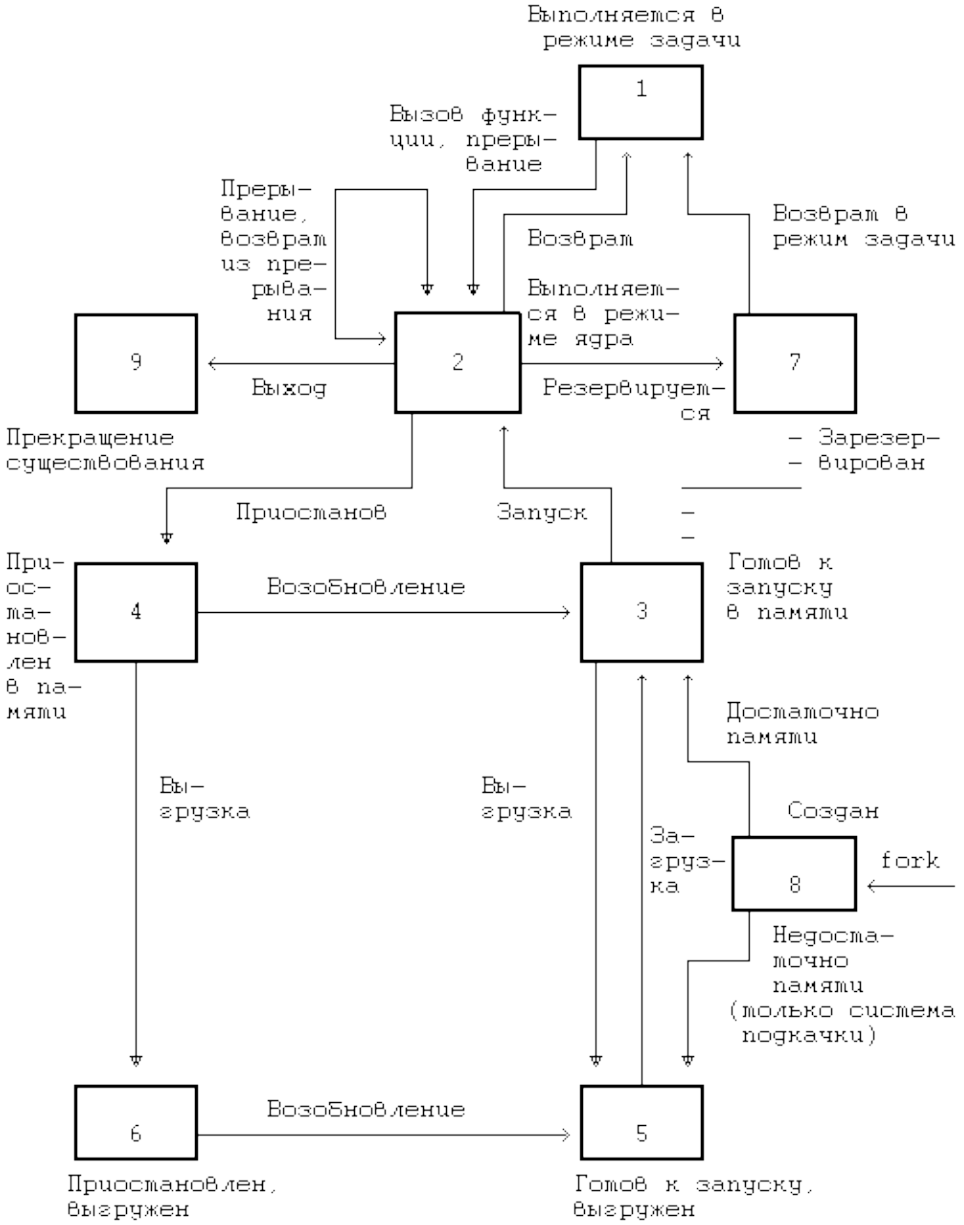


Диаграмма состояний процессов (2)



Просмотр запущенных процессов

$ ps aux

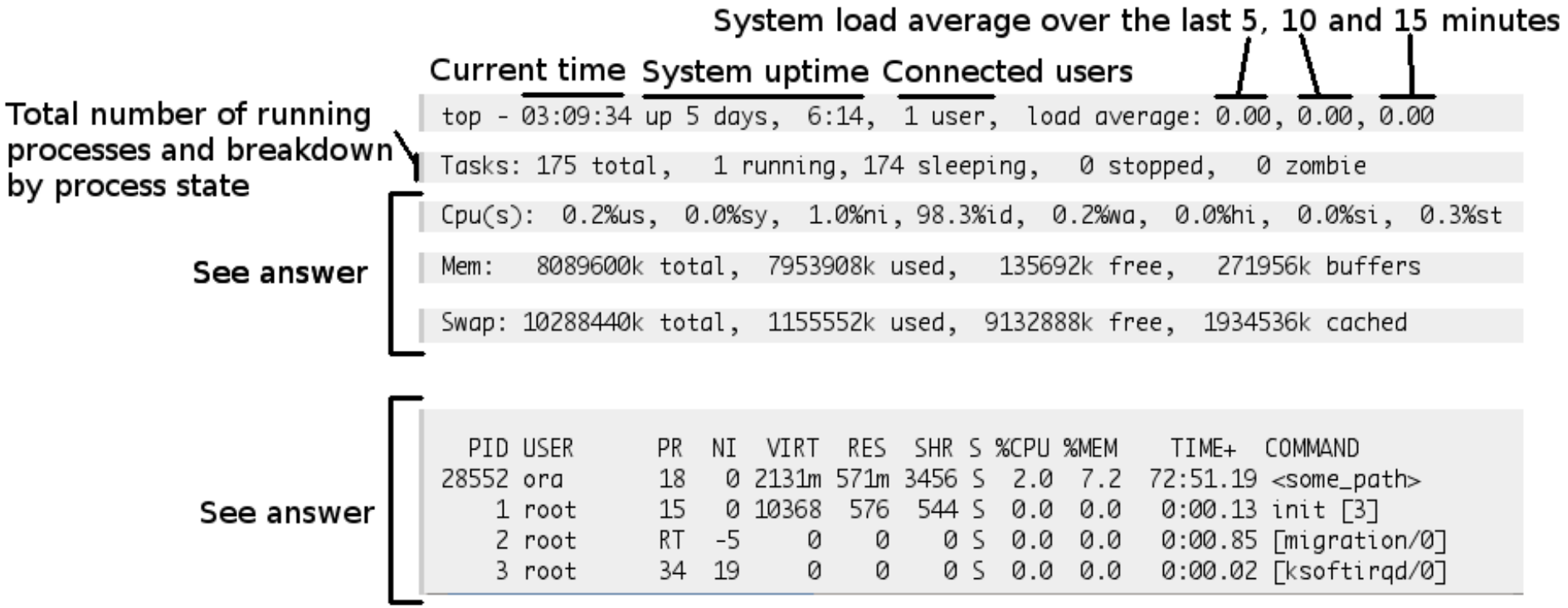
```
daemon    1048  0.0  0.0  26044  2188 ?        Ss   10:50   0:00 /usr/sbin/atd -f
syslog    1055  0.0  0.0 262688  3720 ?        Ssl  10:50   0:01 /usr/sbin/rsyslogd -n
root      1056  0.0  0.0  28628  3136 ?        Ss   10:50   0:00 /lib/systemd/systemd-logind
root      1063  0.0  0.0   4396  1256 ?        Ss   10:50   0:01 /usr/sbin/acpid
```

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
```

- Пользователь, запустивший процесс
- Уникальный идентификатор процесса (PID)
- Процент использования CPU
- Процент использования памяти
- Виртуальная память
- Резидентная память
- Устройство вывода
- Текущий статус процесса
- Время выполнения
- Команда, запустившая процесс

Просмотр top процессов

$ top



https://superuser.com/questions/575202/understanding-top-command-in-unix

Приоритет процесса

Можно повысить или понизить приоритет выполнения процесса.

$ nice <value> -p <pid>

- value - число от -20 до 19 (по-умолчанию - 0)
- Чем ниже value, тем выше приоритет

Остановка процессов

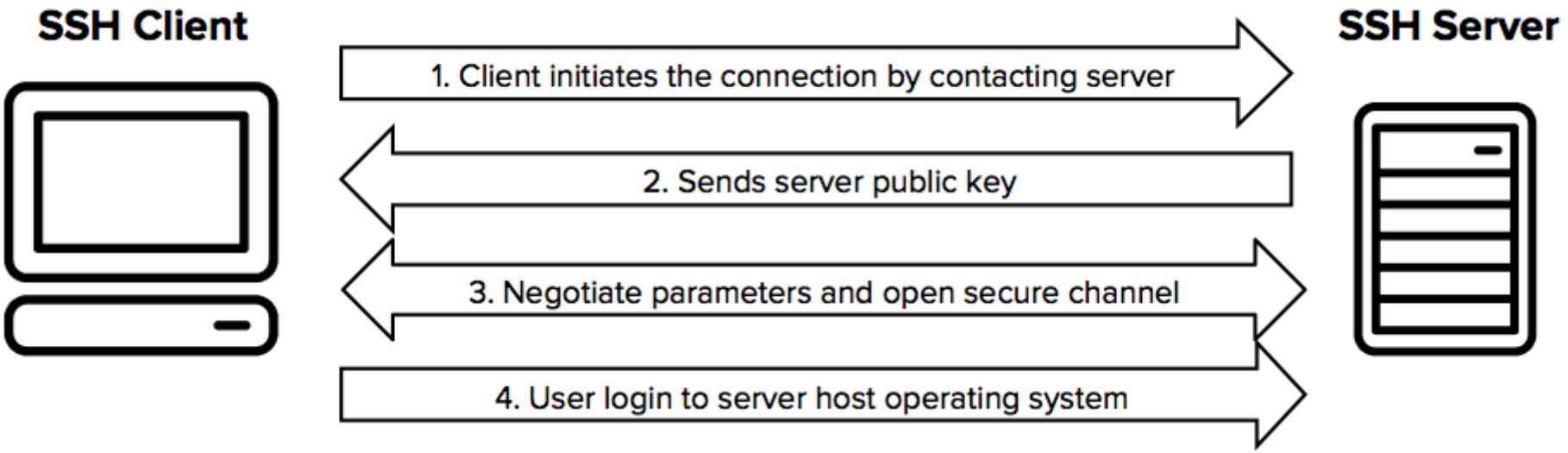
$ kill <pid>
$ kill -9 <pid>

$ pkill <pname>
$ pkill --signal 2 <pname>

SSH

SSH (англ. Secure Shell — «безопасная оболочка»[1]) — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

SSH Scheme


SSH Client
SSH Server
1. Client initiates the connection by contacting server
2. Sends server public key
3. Negotiate parameters and open secure channel
4. User login to server host operating system

ssh login

Подключение к машине по IP адресу:
$ ssh <user>@<host>

Например (по-умолчанию порт 22):
$ ssh user@127.0.0.1

Если настроен нестандартный порт для ssh:
$ ssh <user>@<host> -p 23

ssh keygen

Создание ключа (с настройками по-умолчанию):
$ ssh-keygen

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:X93fz2D1zIH7z1B1ToT1UXk1N3lLVuqrgs9UFsUCxAs user@T8S170705
The key's randomart image is:
```

В папке ~/.ssh/ будет создано два файла:
$ ls -l ~/.ssh

```
-rw------- 1 user user 1675 дек 24 11:09 id_rsa
-rw-r--r-- 1 user user  396 дек 24 11:09 id_rsa.pub
```

id_rsa - приватный/закрытый ключ
id_rsa.pub - публичный/открытый ключ

ssh-copy-id

Для входа на сервер ssh по ключу необходимо скопировать содержимое файла id_rsa.pub в файл ~/.ssh/authorized_keys на сервере

$ cat user.id_rsa.pub >> ~/.ssh/authorized_keys

Теперь пароль для входа не нужен. Для каждой машины, с которой планируется вход, необходимо установить ключ.

Для копирования ключа на удалённую машину есть специальная команда:
$ ssh-copy-id user@host
$ ssh-copy-id -i ~/.ssh/id_rsa.pub user@host

После установки ключа можно запрещать доступ к удалённой машине по паролю (настройка в /etc/ssh/sshd_config).

ssh server

Установка пакета ssh сервера:
$ sudo apt-get install openssh-server

Настройка конфигурационного файла:
$ sudo nano /etc/ssh/sshd_config

Ключевые настройки безопасности:
Port 22 (обычно меняют, если сервер смотрит наружу)
PermitEmptyPasswords no (не разрешаем пустые пароли)
PasswordAuthentication no (запрещаем пароли, вход только по ключу)

После изменения конфигурации нужен перезапуск:
$ sudo /etc/init.d/ssh restart

ssh config

Для удалённых машин можно создать именнованные конфигурации на локальной машине в файле ~/.ssh/config

$ nano ~/.ssh/config

```
Host server1
    HostName 192.168.1.1
    Port 23
    User user
```

Теперь мы можем не вводить каждый раз IP и порт.

$ ssh server1

ssh -t

Допустим, есть доступ к host1, но нет доступа к host2 (он за firewall, например), но доступ к host2 есть у host1. Тогда подключиться к host2 можно так:

$ ssh -t user@host1 ssh user@host2

ssh -X / ssh -Y

При помощи графической системы X Window System и ssh можно пробрасывать графику с удалённой машины на локальную. То есть запускать приложения на удалённой машине, но отображать GUI на локальной.

Запуск программы на удалённой машине:
$ ssh -X user@host firefox

Вход по ssh и возможность последующего запуска:
$ ssh -Y user@host
host $ firefox

Для этой функции в файле настроек ssh сервера /etc/ssh/sshd_config должна быть включена опция:
X11Forwarding yes

SSH port forwarding

SSH можно использовать для построения туннелей к определённым ресурсам на удалённом компьютере.

Пример:
Допустим есть внещний IP, по которому доступен удалённый компьютер. Допустим также, что удалённый компьютер входит в локальную сеть, в которой есть web сервис, доступ к которому мы хотим получить (192.168.1.1:80). Тогда можно создать туннель к этому web сервису на локальный компьютер через внешний IP:

```
$  ssh -f -N -L 8080:192.168.1.1:80 user@host
```

Теперь web сервис будет доступен по адресу:
http://localhost:8080

SSH для неустойчивых соединений

При подключении через неустойчивые каналы ssh по-умолчанию не умеет реагировать и будет обрывать связь. Это потребует постоянного переподключения. Но есть обёртка - autossh.

```
$ sudo apt-get autossh
$ autossh user@host
```

Другая утилита - mosh. Это специально оптимизированная для неустойчивых и низкоскоростных соединений версия SSH, работающая по протоколу UDP. Mosh позволяет получить быстрое и отзывчивое соединение даже на очень медленном канале и из коробки умеет поднимать упавшее соединение и даже переключать клиента с одного IP на другой (при переключении с Wi-Fi-соединения на мобильное, например) без перезапуска сессии (https://mosh.org/)

На клиенте и на сервере необходимо установить mosh, а также проверить, что открыты порты для UDP, требуемые для работы (60000-61000).

```
$ sudo apt-get install mosh
$ mosh user@host
```

Полезные ссылки

Процессы:
http://heap.altlinux.org/tmp/unix_base/ch01s03.html
https://www.ibm.com/developerworks/ru/library/au-speakingunix8/
https://www.ibm.com/developerworks/ru/library/au-unixprocess/index.html
https://www.brianstorti.com/an_introduction_to_unix_processes/

Сигналы:
https://ru.wikipedia.org/wiki/Сигнал_(Unix)

SSH:
https://ru.wikipedia.org/wiki/SSH
https://www.ssh.com/ssh/
https://mosh.org/