Демоны UNIX

Демон - фоновый процесс, не имеющий управляющего терминала.

> Controlling Terminal: A terminal that is associated with a session. Each session may have at most one controlling terminal associated with it, and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal cause signals to be sent to all processes in the foreground process group associated with the controlling terminal.
>
> Session: A collection of process groups established for job control purposes. Each process group is a member of a session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its creator. A process can alter its session membership; see setsid(). There can be multiple process groups in the same session.
>
> Process Group: A collection of processes that permits the signaling of related processes. Each process in the system is a member of a process group that is identified by a process group ID. A newly created process joins the process group of its creator.

POSIX: General Terminal Interface

Статьи от сотрудников Selectel:

- Краткое введение в терминалы и консоль
- Сага о том, как мы писали консоль
- Эмулятор терминала Pyte

nohup(1)

Unix Daemon Server Programming

```
ps -axj
```

Типичные демоны

- syslog: syslog-ng, rsyslog
- cron: ISC cron (бывший Vixie cron), cronie
- CUPS
- dhcpcd
- Apache httpd, Nginx, PostgreSQL, MySQL, Redis, Docker etc.

Алгоритм демонизации

1. Вызвать функцию `umask`, чтобы сбросить маску режима создания файлов в значение `0`.
2. Вызвать функцию `fork` и завершить родительский процесс.
   - заставляем командную оболочку “думать”, что команда выполнилась;
   - дочерний процесс не будет являться лидером группы.
3. Создать новый сеанс, обратившись к функции `setsid`.
   - процесс становится лидером нового сеанса,
   - лидером новой группы процессов,
   - лишается управляющего терминала.
4. Сделать корневой каталог (или каталог, заданный в конфигурации) текущим рабочим каталогом через `chdir`.
5. Закрыть все файловые дескрипторы.
6. Открыть файловые дескрипторы с номерами `0`, 1 и 2 на устройстве `/dev/null`.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <syslog.h>
```

```c
#include <fcntl.h>
#include <unistd.h>
#include <sys/resource.h>
#include <sys/stat.h>
#include <sys/types.h>


void daemonize(const char* cmd)
{
    /*
     * Инициализировать файл журнала.
     */
    openlog(cmd, LOG_CONS, LOG_DAEMON);

    /*
     * Сбросить маску режима создания файла.
     */
    umask(0);

    /*
     * Получить максимально возможный номер дескриптора файла.
     */
    struct rlimit rl;
    if (getrlimit(RLIMIT_NOFILE, &rl) < 0)
        perror("невозможно получить максимальный номер дескриптора");

    /*
     * Стать лидером нового сеанса, чтобы утратить управляющий терминал.
     */
    pid_t pid;
    if ((pid = fork()) < 0)
        perror("ошибка вызова функции fork");
    else if (pid != 0) /* родительский процесс */
        exit(0);
    setsid();

    /*
     * Обеспечить невозможность обретения управляющего терминала в будущем.
     */
    struct sigaction sa;
    sa.sa_handler = SIG_IGN;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    if (sigaction(SIGHUP, &sa, NULL) < 0)
        syslog(LOG_CRIT, "невозможно игнорировать сигнал SIGHUP");

    if ((pid = fork()) < 0)
        syslog(LOG_CRIT, "ошибка вызова функции fork");
    else if (pid != 0) /* родительский процесс */
        exit(0);

    /*
     * Назначить корневой каталог текущим рабочим каталогом,
     * чтобы впоследствии можно было отмонтировать файловую систему.
     */
    if (chdir("/") < 0)
        syslog(LOG_CRIT, "невозможно сделать текущим рабочим каталогом /");

    /*
     * Закрыть все открытые файловые дескрипторы.
     */
```

```
    if (rl.rlim_max == RLIM_INFINITY)
        rl.rlim_max = 1024;
    for (int i = 0; i < rl.rlim_max; i++)
        close(i);

    /*
     * Присоединить файловые дескрипторы 0, 1 и 2 к /dev/null.
     */
    int fd0 = open("/dev/null", O_RDWR);
    int fd1 = dup(0);
    int fd2 = dup(0);
    if (fd0 != 0 || fd1 != 1 || fd2 != 2)
        syslog(LOG_CRIT, "ошибочные файловые дескрипторы %d %d %d",
               fd0, fd1, fd2);
}
```

```
$ ./a.out
$ ps -axj
PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
   1 24987 24986 24986 ?           -1 S     1000   0:00 ./a.out
$ ps -axj | grep 24986
PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
   1 24987 24986 24986 ?           -1 S     1000   0:00 ./a.out
```

Why setsid?

1. The process becomes a session leader of a new session that only contains the calling process. (PID = SID)
2. The process becomes the process group leader of a new group. (PID = SID = PGID)
3. The process will have no controlling terminal. If it had one before setsid(), the association will be broken.

Why fork() again?

1. In order for the process to be reparented to init, its parent must exit.
2. The second call to fork() prevents the daemon from ever acquiring a controlling terminal again.

Why POSIX Daemonization is Complicated, перевод

Журналирование: syslog

syslog(3)

```
#include <syslog.h>

void openlog(const char* ident, int option, int facility);
void syslog(int priority, const char* format, ...);
void closelog(void);

/* option */
LOG_CONS;
LOG_NDELAY;
LOG_NOWAIT;
LOG_ODELAY;
LOG_PERROR;
LOG_PID;

/* facility */
LOG_AUTH;
LOG_AUTHPRIV;
LOG_CRON;
LOG_DAEMON;
LOG_FTP;
```

```
LOG_KERN;
LOG_LOCAL0 - LOG_LOCAL7;
LOG_LPR;
LOG_MAIL;
LOG_NEWS;
LOG_SYSLOG;
LOG_USER;
LOG_UUCP;

/* priority */
LOG_EMERG;
LOG_ALERT;
LOG_CRIT;
LOG_ERR;
LOG_WARNING;
LOG_NOTICE;
LOG_INFO;
LOG_DEBUG;
```

```
openlog("lpd", LOG_PID, LOG_LPR);
syslog(LOG_ERR, "open error for %s: %m", filename);
```

- https://loggly.com
- https://papertrail.com

Журналирование: библиотеки

zlog

```
int main(int argc, char** argv)
{
    int rc;
    zlog_category_t *c;

    rc = zlog_init("/etc/zlog.conf");
    if (rc) {
        printf("init failed\n");
        return -1;
    }

    c = zlog_get_category("my_cat");
    if (!c) {
        printf("get cat fail\n");
        zlog_fini();
        return -2;
    }

    zlog_info(c, "hello, zlog");

    // zlog_fatal
    // zlog_error
    // zlog_warn
    // zlog_notice
    // zlog_info
    // zlog_debug

    zlog_fini();

    return 0;
}
```

libU

GLib

Конфигурация

- Conf: libconfuse, libconfig, libucl
- INI: GLib, iniparser, libconfini
- JSON: C/C++ JSON parser/generator benchmark; jzon
- XML: libxml2, GLib
- YAML: libYAML
- SQL: SQLite
- Scripting: Lua, Python, Duktape, QuickJS

Приоритет конфигураций

1. Аргументы командной строки
2. Переменные окружения
3. Локальный (пользовательский) файл конфигурации
4. Глобальный (системный) файл конфигурации
5. Умолчания; “sensible defaults”

Запуск в единственном экземпляре

fcntl(2)

```
#define LOCKFILE "/var/run/my-daemon.pid"
#define LOCKMODE (S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH)

int lockfile(int fd)
{
    struct flock fl;
    fl.l_type = F_WRLCK;
    fl.l_start = 0;
    fl.l_whence = SEEK_SET;
    fl.l_len = 0;
    return(fcntl(fd, F_SETLK, &fl));
}

int already_running(void)
{
    int fd;
    char buf[16];

    fd = open(LOCKFILE, O_RDWR|O_CREAT, LOCKMODE);
    if (fd < 0) {
        syslog(LOG_ERR, "невозможно открыть %s: %s",
               LOCKFILE, strerror(errno));
        exit(1);
    }
    if (lockfile(fd) < 0) {
        if (errno == EACCES || errno == EAGAIN) {
            close(fd);
            return 1;
        }
        syslog(LOG_ERR, "невозможно установить блокировку на %s: %s",
               LOCKFILE, strerror(errno));
        exit(1);
    }
    ftruncate(fd, 0);
    sprintf(buf, "%ld", (long)getpid());
    write(fd, buf, strlen(buf)+1);
```

```
    return 0;
}
```

Запуск

Системы инициализации

Comparison of init systems

- GNU sysvinit
- Upstart
- OpenRC
- systemd
- BSD init(8)
- Busybox
- runit
- daemontools
- s6

```
#! /bin/sh

### BEGIN INIT INFO
# Provides:          nginx
# Required-Start:    $all
# Required-Stop:     $all
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: starts the nginx web server
# Description:       starts nginx using start-stop-daemon
### END INIT INFO

PATH=/opt/bin:/opt/sbin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/opt/sbin/nginx
NAME=nginx
DESC=nginx

test -x $DAEMON || exit 0

# Include nginx defaults if available
if [ -f /etc/default/nginx ] ; then
        . /etc/default/nginx
fi

set -e

case "$1" in
  start)
        echo -n "Starting $DESC: "
        start-stop-daemon --start --quiet --pidfile /var/run/nginx.pid \
                --exec $DAEMON -- $DAEMON_OPTS
        echo "$NAME."
        ;;
  stop)
        echo -n "Stopping $DESC: "
        start-stop-daemon --stop --quiet --pidfile /var/run/nginx.pid \
                --exec $DAEMON
        echo "$NAME."
        ;;
  restart|force-reload)
        echo -n "Restarting $DESC: "
        start-stop-daemon --stop --quiet --pidfile \
```

```
                /var/run/nginx.pid --exec $DAEMON
        sleep 1
        start-stop-daemon --start --quiet --pidfile \
                /var/run/nginx.pid --exec $DAEMON -- $DAEMON_OPTS
        echo "$NAME."
        ;;
  reload)
      echo -n "Reloading $DESC configuration: "
      start-stop-daemon --stop --signal HUP --quiet --pidfile /var/run/nginx.pid \
          --exec $DAEMON
      echo "$NAME."
      ;;
  *)
        N=/etc/init.d/$NAME
        echo "Usage: $N {start|stop|restart|force-reload}" >&2
        exit 1
        ;;
esac

exit 0
```

start-stop-daemon(8)

```
#!/sbin/openrc-run
# Copyright 1999-2017 Gentoo Foundation
# Distributed under the terms of the GNU General Public License v2

extra_commands="configtest"
extra_started_commands="upgrade reload"

description="Robust, small and high performance http and reverse proxy server"
description_configtest="Run nginx' internal config check."
description_upgrade="Upgrade the nginx binary without losing connections."
description_reload="Reload the nginx configuration without losing connections."

NGINX_CONFIGFILE=${NGINX_CONFIGFILE:-/etc/nginx/nginx.conf}

command="/usr/sbin/nginx"
command_args="-c \"${NGINX_CONFIGFILE}\""
start_stop_daemon_args=${NGINX_SSDARGS:-"--wait 1000"}
pidfile=${NGINX_PIDFILE:-/run/nginx.pid}
user=${NGINX_USER:-nginx}
group=${NGINX_GROUP:-nginx}
retry=${NGINX_TERMTIMEOUT:-"TERM/60/KILL/5"}

depend() {
    need net
    use dns logger netmount
}

start_pre() {
    if [ "${RC_CMD}" != "restart" ]; then
        configtest || return 1
    fi
}

stop_pre() {
    if [ "${RC_CMD}" = "restart" ]; then
        configtest || return 1
    fi
}
```

```
stop_post() {
    rm -f ${pidfile}
}

reload() {
    configtest || return 1
    ebegin "Refreshing nginx' configuration"
    start-stop-daemon --signal SIGHUP --pidfile "${pidfile}"
    eend $? "Failed to reload nginx"
}

# ...

configtest() {
    ebegin "Checking nginx' configuration"
    ${command} -c "${NGINX_CONFIGFILE}" -t -q

    if [ $? -ne 0 ]; then
        ${command} -c "${NGINX_CONFIGFILE}" -t
    fi

    eend $? "failed, please correct errors above"
}
```

```
$ /etc/init.d/nginx
Usage: nginx [options] stop | start | restart | describe | zap

Options: [ dDsSvl:ZChqVv ]
  -d, --debug                       set xtrace when running the script
  -Z, --dry-run                     show what would be done
  -s, --ifstarted                   only run commands when started
  -S, --ifstopped                   only run commands when stopped
  -D, --nodeps                      ignore dependencies
  -l, --lockfd <arg>                fd of the exclusive lock from rc
  -h, --help                        Display this help output
  -C, --nocolor                     Disable color output
  -V, --version                     Display software version
  -v, --verbose                     Run verbosely
  -q, --quiet                       Run quietly (repeat to suppress errors)
```

```
[Unit]
Description=The nginx HTTP and reverse proxy server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecStartPost=/bin/sleep 0.1
ExecReload=/bin/kill -HUP $MAINPID
ExecStop=/bin/kill -QUIT $MAINPID

[Install]
WantedBy=multi-user.target
```

Systemd за пять минут

New-Style Daemons

Broken by design: systemd

Systemd invasion into Linux Server space RHEL 7 as Red Hat fiasco

https://ru.wikipedia.org/wiki/CoreOS

Daemon best practices

- Файл блокировки в `/var/run/daemon-name.pid`
- Файл конфигурации в `/etc/daemon-name.conf`
- Запуск через `/etc/init.d/daemon-name`, возможность запуска без демонизации
- Перечитывание конфигурации по `SIGHUP`

Библиотеки демонизации

daemon(3)

> Not in POSIX.1. A similar function appears on the BSDs. The daemon() function first appeared in 4.4BSD.
>
> The GNU C library implementation of this function was taken from BSD, and does not employ the double-fork technique (i.e., fork(2), setsid(2), fork(2)) that is necessary to ensure that the resulting daemon process is not a session leader. Instead, the resulting daemon is a session leader. On systems that follow System V semantics (e.g., Linux), this means that if the daemon opens a terminal that is not already a controlling terminal for another session, then that terminal will inadvertently become the controlling terminal for the daemon.

Daemonize - a Tiny C Library for Programming the UNIX Daemons

Simple example of daemon for Linux