**Liunx 信号**

1. liunx 中文手册安装

查考：<https://avdancedu.com/a77533/>

<https://blog.csdn.net/monkeyhi/article/details/105317717>

（1）安装系统依赖

$  brew install python3

$ brew install autoconf

$ brew install automake

$ brew install opencc

(2) 下载中文手册包，编译安装

cd manpages-zh

autoreconf --install --force

./configure

make

sudo make install

（3）在/etc/man.conf修改和添加

sudo vim /etc/man.conf

...

MANPATH /usr/local/X11/man #line 44

MANPATH /usr/local/share/man/zh\_CN #添加中文手册路径

...

:wq

1. 解决乱码,修改man.conf

$ brew install groff

$ sudo vim /etc/man.conf

...

#修改NROFF配置如下（将UTF8编码的MAN页面通过转码而被groff识别）

#line 95

NROFF preconv -e utf8 | /usr/local/bin/groff -Wall -mtty-char -Tutf8 -mandoc -c

...

#修改PAGER配置如下（这样可以避免MAN手册页面中的ANSI Escape字符序列干扰（用于控制显示粗体等格式））

#line 106

PAGER /usr/bin/less -isR

...

:wq

(5)最后 source /etc/man.conf 生效一下

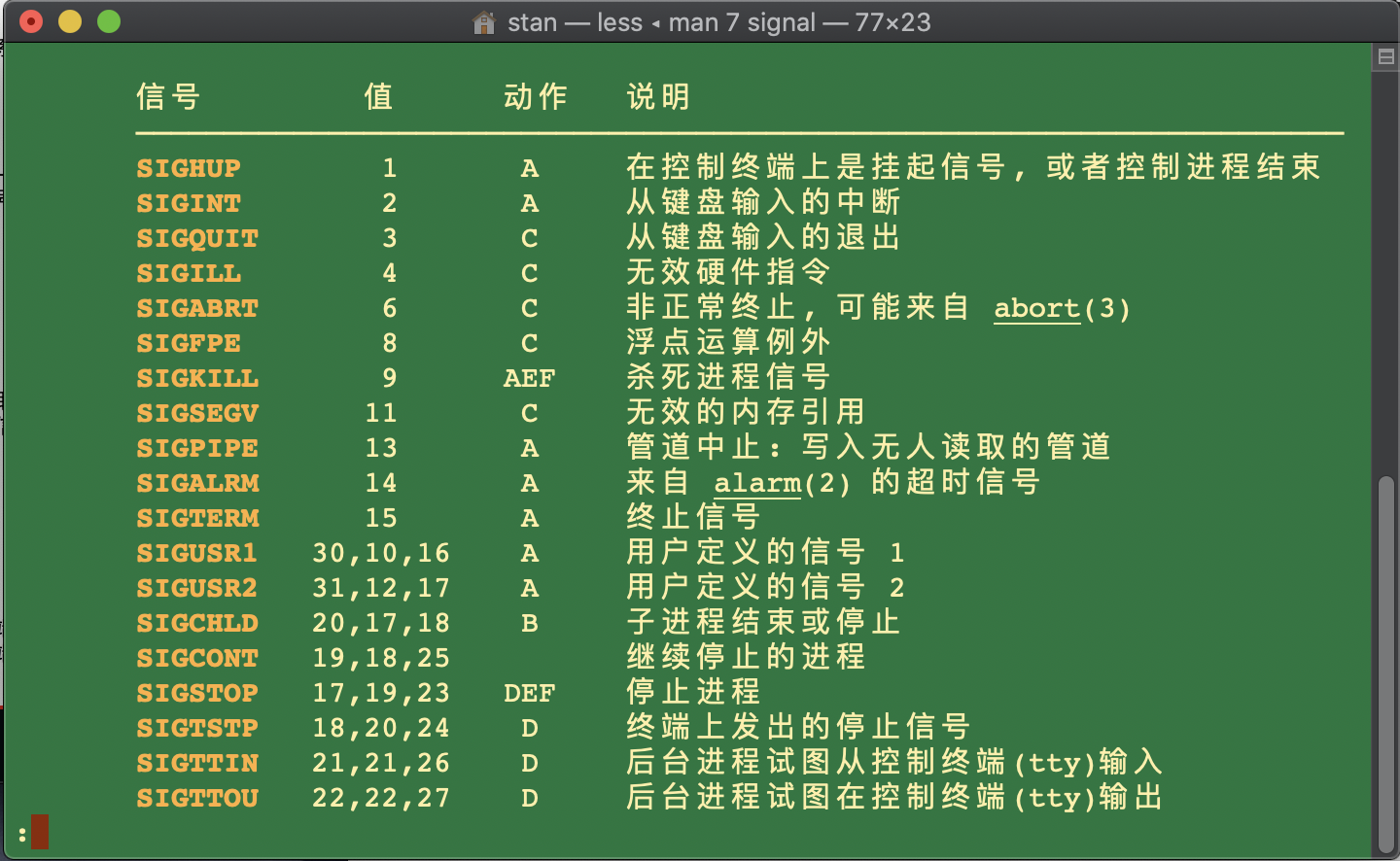
2.liunx 信号

（1）Liunx 运行中，中断会发出信号，这就叫信号，

（2）信号的处理方式：忽略，捕获，默认处理

（3）查看liunx下信号：man 7 signal

（4）如下图

****

说明：信号是宏定义的以SIG开头如果动作是A，进程默认会被杀死，不能捕获AEF/DEF

这系统捕获，如果我们发送AEF/DEF信号，进程就会被杀死。

A 缺省动作是结束进程.

B 缺省动作是忽略这个信号.

C 缺省动作是结束进程，并且核心转储.

D 缺省动作是停止进程.

E 信号不能被捕获.

SIGHUP 程序被杀死后，收到这个信号，系统会杀死这个程序所在到进程

SIGINT 键盘输入control+/ 终止进程

SIGQUIT 比如我们用control+c 系统就会收到这个信号，把进程终止

（5）几个和重要的信号

SIGPIPE管道中止,当写入无人读取的管道时产生该信号,默认终止进程

SIGCHLD子进程结束或停止时发送

SIGALRM定时器信号,以秒为单位，默认终止进程

SIGUSR1/SIGUSR2自定义,默认终止进程

SIGINT键盘输入的退出信号 cotr+\

SIGQUIT键盘输入的退出信号 cotr+c

SIGHUP控制终端的挂起信号

SIGPIPE:网络程序必须要处理SIGPIPE信号,否则当客户端

退出后,服务器仍然向该SOCKET发数据时，则会

引起Crash

SIGCHLD:1）僵尸进程是-个早已死亡的进程,但在进程表中仍占有位置,

僵尸进程，它的其他资源已经释放了，但是在进程表中还是

占用了位置，它会一直占用着。因为系统的进程数量是一定

的，这样回对系统造成影响。

2）Linux中当子进程结束的时候,他并没有被完全销毁,因为父进程还

要用它的信息统计。

3）父进程没有处理SIGCHLD信号或调用wait / waitpid()等待子进

程结束,就会出现僵尸进程

（6）信号的发送

硬件方式 ctrl + c ctrl + \

软件方式 kill api

(7) 信号的安装

简单方式 signal(int sig, void (\*func)(int)); 第一个参数是要捕获的系统信号，第二个参数是回调函数

高级方式int sigaction(int sig,

const struct sigaction\* act,

struct sigaction\* oact);

第一个参数是要捕获的信号

第二个参数是结构体包含回调的参数，和flag

第三个参数是老的action，一般设置null

struct sigaction {

void(\*sa\_handler)(int); //处理函数

void(\*sa\_sigaction)(int, siginfo\_t\*,void\*); //处理函数 一般不用了解

sigset\_t sa\_mask; //掩码 屏蔽哪些信号

int sa\_flags; //根据SA\_SIGINFO标记选择sa\_handler/sa\_sigaction

void(\*sa restorer)(void); // 设置NULL,一般不用

};

1. 进程的创建

<1>fork，和调用系统调 daemon API创建,

<2>fork一个子进程，父进程退出，子进程成为孤儿进程，被init进程接管

<3>调用setsid创建新进程会话，因为现在进程标记的还是，以前的父进程

<4>切换当前的工作目录到根目录

<5>将标准输入，输出，出错重定向到/dev/null