目录

[一：重要软件下载地址 2](#_Toc533237743)

[二：gcc /g++命令重要参数 2](#_Toc533237744)

[三：信号相关 2](#_Toc533237745)

# 一：重要软件下载地址

链接：https://pan.baidu.com/s/147TP-jTHad3-Trfx1wC-Rg

提取码：46yn

# 二：gcc /g++命令重要参数

(1)-o：指定编译链接后生成的可执行文件名，比如

gcc -o nginx nginx.c

(2)-c: 将.c编译成.o目标文件[仅执行编译操作，不进行链接操作]

gcc -c nginx.c

将生成nginx.o的目标文件

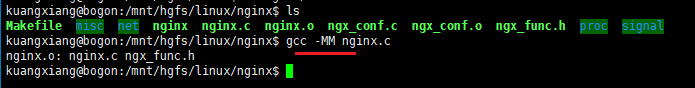
(3)-M：显示一个源文件所依赖的各种文件

gcc -M nginx.c

(4)-MM：显示一个源文件所依赖的各种文件，但不包括系统的一些头文件；

gcc -MM nginx.c

这种扫描是有用途的，尤其是在写makefile文件时，需要用到这些依赖关系，以做到比如当某个.h头文件更改时，整个工程会实现自动重新编译的目的；



(5) -g：生成调试信息。GNU 调试器可利用该信息。

(6)-I：gcc会先到你 用这个参数指定的目录去查找头文件，你在.c/.cpp中可以

#include <abc.h> //这里用尖括号了

gcc -I /mnt/mydir

# 三：信号相关

kill 命令不同数字所能发出的不同信号

|  |  |  |
| --- | --- | --- |
| kill的参数 | 该参数发出的信号 | 操作系统缺省动作 |
| -1 | SIGHUP（连接断开） | 终止掉进程（进程没了） |
| -2 | SIGINT（终端中断符，比如ctrl+c） | 终止掉进程（进程没了） |
| -3 | SIGQUIT（终端退出符，比如ctrl+\） | 终止掉进程（进程没了） |
| -9 | **SIGKILL（终止）** | 终止掉进程（进程没了） |
| -18 | SIGCONT（使暂停的进程继续） | 忽略（进程依旧在运行不受影响） |
| -19 | **SIGSTOP（停止），可用SIGCONT继续，但任务被放到了后台** | 停止进程（**不是终止，进程还在**） |
| -20 | SIGTSTP（终端停止符，比如ctrl+z），但任务被放到了后台，可用SIGCONT继续 | 停止进程（**不是终止，进程还在**） |

进程状态：

|  |  |
| --- | --- |
| 状态 | 含义 |
| D | 不可中断的休眠状态(通常是I/O的进程)，可以处理信号，有 延迟 |
| R | 可执行状态&运行状态(在运行队列里的状态) |
| S | 可中断的休眠状态之中（等待某事件完成），可以处理信号 |
| T | 停止或被追踪（被作业控制信号所停止） |
| Z | 僵尸进程 |
| X | 死掉的进程 |
| < | 高优先级的进程 |
| N | 低优先级的进程 |
| L | 有些页被锁进内存 |
| s | Session leader（进程的领导者），在它下面有子进程 |
| t | 追踪期间被调试器所停止 |
| + | 位于前台的进程组 |

常用信号列举：

|  |  |
| --- | --- |
| 信号名 | 信号含义 |
| SIGHUP（连接断开） | 是终端断开信号，如果终端接口检测到一个连接断开，发送此信号到该终端所在的会话首进程（前面讲过），缺省动作会导致所有相关的进程退出(上节课也重点讲了这个信号，xshell断开就有这个信号送过来)；  Kill -1 进程号也能发送此信号给进程； |
| SIGALRM（定时器超时） | 一般调用系统函数alarm创建定时器，定时器超时了就会这个信号； |
| SIGINT（中断） | 从键盘上输入ctrl+C（中断键）【比如你进程正跑着循环干一个事】，这一ctrl+C就能打断你干的事，终止进程；  **但shell会将后台进程对该信号的处理设置为忽略（也就是说该进程若在后台运行则不会收到该信号）**； |
| SIGSEGV（无效内存） | 内存访问异常，除数为0等，硬件会检测到并通知内核；其实这个SEGV代表段违例（segmentation violation），你有的时候运行一个你编译出来的可执行的c程序，如果内存有问题，执行的时候就会出现这个提示； |
| SIGIO（异步I/O） | 通用异步I/O信号，咱们以后学通讯的时候，如果通讯套接口上有数据到达，或发生一些异步错误，内核就会通知我们这个信号； |
| SIGCHLD（子进程改变） | 一个进程终止或者停止时，这个信号会被发送给父进程；（我们想象下nginx，worker进程终止时 master进程应该会收到内核发出的针对该信号的通知）； |
| SIGUSR1,SIGUSR2（都是用户定义信号） | 用户定义的信号，可用于应用程序，用到再说； |
| SIGTERM（终止） | 一般你通过在命令行上输入kill命令来杀一个进程的时候就会触发这个信号，收到这个信号后，你有机会退出前的处理，实现这种所谓优雅退出的效果； |
| SIGKILL（终止） | 不能被忽略，这是杀死任意进程的可靠方法，不能被进程本身捕捉 |
| SIGSTOP（停止） | 不能被忽略，使进程停止运行，可以用SIGCONT继续运行，但进程被放入到了后台 |
| SIGQUIT（终端退出符） | 从键盘上按ctrl+\  **但shell会将后台进程对该信号的处理设置为忽略（也就是说该进程若在后台运行则不会收到该信号）**； |
| SIGCONT（使暂停进程继续） | 使暂停的进程继续运行 |
| SIGTSTP（终端停止符） | 从键盘上按ctrl+z，进程被停止，并被放入后台，可以用SIGCONT继续运行 |