信号量不是信号，信号不是信号量----这两个是完全不同的东西...
进程信号:
作用:通知别人，发生了某件事情，尽快的去处理这件事情(操作系统通知进程发生了某个事件，打断进程当前操作，去处理这个事件)
是什么:软件中断
信号想要成为一个中断，首先我们必须认识这个信号,并且知道如何去处理它
事件多种多样，因此信号也是多种多样

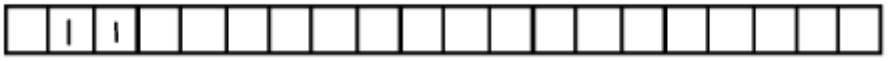
查看操作系统中定义好的信号:使用kil -l命令可以查看信号种类:
用户所能看到并使用的信号共有62种，两大分类:
1~31号信号:每个信号都有具体对应的事件---- 非可靠信号
34~64号信号:在操作系统中当前就没有具体对应的事件了，因此命名也稍微草率一点---- 可靠信号

信号的生命周期:信号的产生->信号在进程中注册->信号在进程中注销->处理信号　　信号的阻塞
信号的产生:
硬件: ctrl+c ctrl+z
软件:
命令: kill -signum pid
kill命令能够杀死一个进程，主要是因为ill命令功能是向进程发送一个信号，默认发送的15号终止信号函数:
int kl(pid_t pid, int sig);向指定的进程发送指定的信号
int raise(int sig):向调用进程自身发送指定的信号
void abort(void);向调用进程自身发送SIGABRT信号,使-个异常的进程退出
unsigned int alarm(unsigned int seconds); seconed秒之后给调用进程发送一个SIGALRM信号， 告诉进程时间到了。

core dump:程序异常退出时，操作系统会保存这个进程的运行信息，便于这个进程的事后调试(默认是关闭的)
ulimit -C 1024　开启core dump,并将core文件最大大小设置为1024kb
gdb ./loop. test　->　core- file core.pid(产生的core文件) - >常规的gdb调试步骤

信号在进程中的注册:让进程知道自己收到了这么一个信号

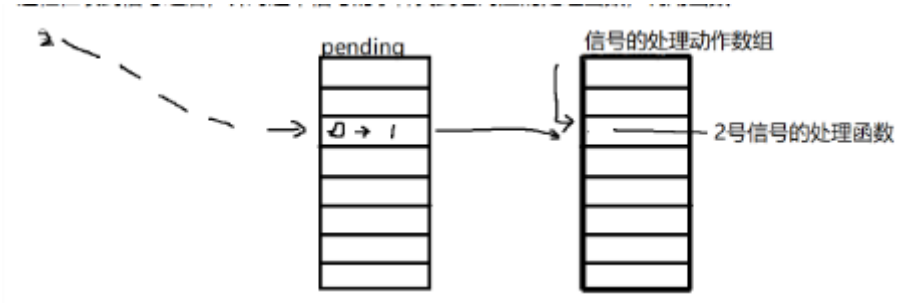
进程就是一个pcb, linux下是一 个task struct结构体;在pcb结构体中定义了一个信号的集合(位图) ;
若给进程发送一个信号， 则将这个信号对应位置的二进制位置1,进程通过查看位图判断是否有信号到来，进而去处理



pending信号集合:未决信号集合:
未决是一个状态，指的是信号产生了，但是还没有被处理的一一个区间状态
这个位图实际上是-个sigset t{unsigned long int_ _val[SIGSET. NWORDS]}结构体;这个数组被用于实现位图
位图这个信号集合，只能用于标记进程是否收到了这个信号,无法确定这个信号收到了多少次
因此在内核中其实还有一个链表 sigqueue{....siginfo_t...}
信号的注册，就是组织一个信号的信息， 添加到信号链表中，并且将信号pending位图进行置位
非可靠信号的注册:在注册信号时，判断当前信号是否已经注册过(位图是否已经为1) ;若没有注册，则添加节点，修改位图;反之，若已经注册过了，则什么也不干(这种信号在链表中就永远顶多只有一个节点,后来到达的信号就会被丢弃 _事件丢失)
可靠信号的注册:在注册信号时，每次针对新到来的信号都会创建一个节点添加到链表中，并且位图置1; (链表中有可能有多 个相同节点)

信号在进程中的注销:抹除信号存在痕迹
非可靠信号:删除节点，位图置0 (因为非可靠信号只会注册一次，顶多只有一个节点)
可靠信号:删除节点，判断是否还有相同节点，若没有，则位图置0;表示没有这个信号了，反之则位图依然为1;表示还有信号待处理

信号的处理:信号的递达
进程处理一个事件，事件可以理解为就是一一个功能，在程序中-一个功能的实现单位就是-一个函数进程在收到信号之后，针对这个信号的事件找到它对应的处理函数，调用函数



在操作系统中，每个信号都有其已经定义好的默认处理动作---信号的默认处理方式
信号的处理方式:
默认处理---系统中已经定义好的默认函数
忽略处理--处理的动作就是什么都不做(依然能够信号注册，只是处理动作中什么也不做而已)
自定义处理-- -用户自己定义信号回调函数，然后使用这个函数的地址替换原有信号动作数组中的函数地址也就是说，替换了信号处理动作中的回调函数