1.

Ctrl-a 回到第一个光标

Ctrl-e 到最后

Ctrl-u 删除所有输入

上 Ctrl-p 上一条指令

下 Ctrl-n 下一条指令

Ctrl-c 终止进程

连续按两次gg回到开头

按G回到末尾

2.

sudo apt-get update 更新软件资源到本地

sudo apt-get install “软件” 安装命令

sudo apt-get remove “软件” 卸载命令

Ubuntu安装包以.deb为结尾 （使用软件包安装 sudo dpkg -i 安装包名）

tar压缩：

tar zcvf 要生成的压缩包名 压缩材料

（tar zcvf test.tar.gz file1 dir2使用gzip方式压缩

tar jcvf test.tar.gz file1 dir2使用bzip2方式压缩

）

tar解压缩：

tar zxvf test.tar.gz使用gzip方式解压缩

tar jxvf test.tar.gz使用bzip2方式解压缩）

rar压缩：

rar a -r 要生成的压缩包名 压缩材料

(

rar a -r test.rar file1 dir2使用rar方式压缩

)

rar 解压：

unrar x rartest.rar

zip压缩：

zip -r testzip.zip file1 dir2

zip解压缩：

unzip testzip.zip

./当前目录

cd ~回到用户目录

history 查看历史命令

pwd 查看当前目录

exit 退出用户

mkdir “文件夹名” 创建文件夹

rmdir 删除空文件夹

rm –rf 强制删除文件

cp 文件/目录 文件/目录 将文件拷贝到目录里

touch file\_name(s) 创建新的空文件

which 查看命令所在的目录

cat 查看文件内容

tac 倒着查看文件内容

tree 以树状显示当前目录

mv 文件/目录 文件/目录 移动文件

whoami 查看当前用户

sudo chmod u+x a.c修改文件权限 (文字设定法)

sudo chmod 471 a.c (数字设定法)

{

u 表示用户

g 表示同组

o 表示其他用户

a 表示所有用户

r 对应4

w 对应2

x 对应1

}

sudo chown 新用户名 待修改文件 修改文件所属用户

sudo chgrp 新用户组名 待修改文件 修改文件所属用户组

sudo chown nobody:nogroup hello.cpp 一起改

find ./ -type ‘l’ 在当前目录下按文件类型找 （找文件）

find ./ -name ‘\*.jpg’ 在当前目录下按文件名字找

find ./ -maxdepth 1 -name ‘\*.jpg’ （限制层级深度为一层，应写为第一个参数）

（还有-size按大小）

grep –r ‘copy’ ./ -n（按文件内容搜索对象）

ps aux 查看当前执行的进程 (ps aux | grep ‘root’ ,两个命令结合使用)

3

bin：存放二进制可执行文件

boot：存放开机启动程序

dev：存放设备文件

home：存放普通用户

etc：用户信息和系统配置文件

lib：库文件

root：管理员宿主目录

usr：用户资源管理目录

4.

-l 详细信息

-a 显示隐藏文件

-d 显示目录

-a 全部

-r 递归

例子：ls –l hello.cpp

5.

Linux系统文件类型：8种

普通文件：-

目录文件：d

字符设备文件：c

块设备文件：b

软连接：l

管道文件：p

套接字：s

未知文件

6.

软连接：为保证软连接可以任意搬移，创建是务必对源文件使用绝对路径

ln –s “文件目录” “软连接名字” 创建软连接也就相当于Windows快捷方式

硬连接：操作系统给每一个文件赋予唯一的inode，当有相同inode的文件存在时，彼此同步。删除时只讲硬连接数减一，减为0时inode被释放

ln “文件目录” “软连接名字” 创建硬连接 文件之间同步

6.

makefile:

命名：makefile Makefile

1个规则，

目标：依赖条件

（一个他把缩进）命令

1. 目标的时间必须晚于依赖条件的时间，否则，更新目录
2. 依赖条件如果不存在，找寻新的规则去产生依赖

ALL:指定makefile的终极目标

2个函数： ($()相当于取值)

src=$(wildcard \*.c):匹配当前工作目录下的所有.c文件。将文件名组成列表，赋值给变量src。

obj=$(patsubst %.c,%.o,$(src))：将参数3中，包含参数1的部分，替换为参数2.

clean:（没有依赖）

-rm -rf $(obj) a.out “-“：作用是，删除不存在文件时，不报错

3个自动变量：

$@:在规则的命令中，表示规则的目标

$^:在规则的命令中，表示所有依赖条件

$<:在规则的命令中，表示第一个依赖条件,如果将该变量应用在模式规则中，它可将依赖条件列表中的依赖依次取出，套用模式规则。

模式规则：

%.o:%.c

gcc –c $< –o $@

静态模式规则：

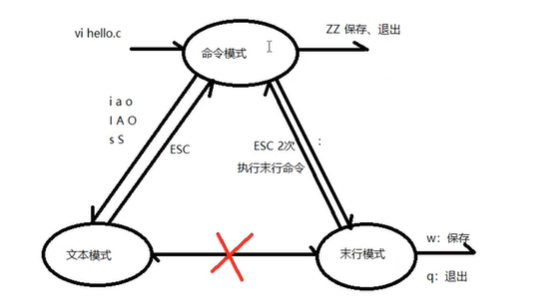
$(obj):%.o:%.c

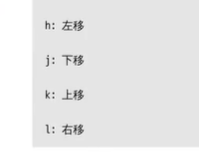
Gcc –c $< -o %@

伪目标：

.PHONY:clean ALL

7.





跳转到指定行：

1）88G(命令模式)

2）:88（末行模式）

3）连续按两次gg回到开头

4）按G回到末尾

5）自动格式化 gg=G（命令模式）

6）删除单个字符 x（命令模式）

7）删除光标至行尾D

8) 删除光标至行尾d0

9）复制一行 yy

10）粘贴 p：向后 P：向前

11）查找 1找设想内容，按“/“输入欲搜索关键字，回车，使用n检索下一个

2找看到的内容，将光标置于单词任意一个字符上，按“\*“

12）撤销 u（命令模式）

13）竖分屏 vsp（末行模式） Crtl+ww切换

8.gcc编译可以执行程序4步骤：预处理、编译。汇编。链接



-I 指定头文件目录位置

-c 只做预处理、编译、汇编 得到二进制文件

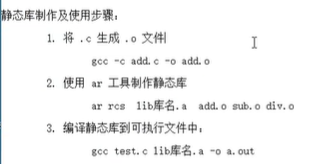
-g 编译是增加调试语句。主要支持gdb调试

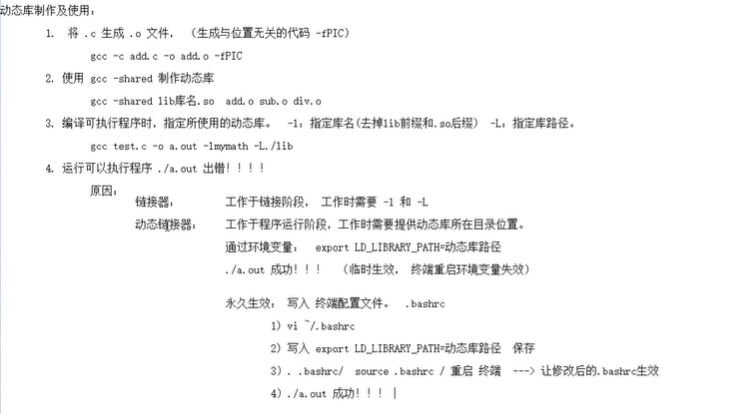
-Wall 显示所有警告信息

-D 向程序动态注册宏定义

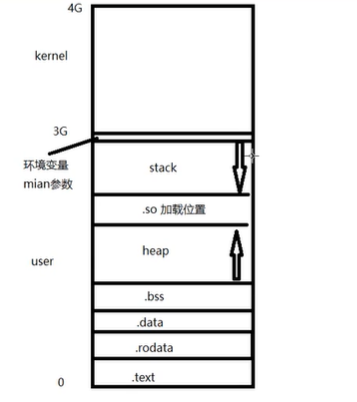
9.静态库：对空间要求较低。而对时间要求较高。动态库：对时间要求较低，对空间要求较高。

编译阶段出错有行号，链接阶段出错没有行号，且有collect2（链接器）





10.进程的内存分布



11.gdb调试工具：

基础指令：

-g：使用该参数编译可执行文件，得到调试表

gdb a.out

list: list/l 1 列出源码。根据源码指定行号设置断点

b：break/b 20 在20行处设置断点

d：delete/d 20 删除某个断点

run/r 运行程序

next/n 下一条指令（会越过函数）

step/s 下一条指令（会进入函数）

print/p p i 查看变量i的值

continue：继续执行断点后续程序

quit 退出gdb当前调试

其他指令：

run：进入gdb直接run就可以查看段错误

set args 设置main 函数命令行参数

run 参数1 参数2 设置main 函数命令行参数

start:直接从第一行开始进行调试

finsh:结束当前函数的调用

info b 查看设置的断点位置

b 20 if i = 5 ：设置条件断点

ptype arr:查看arr变量的变量类型

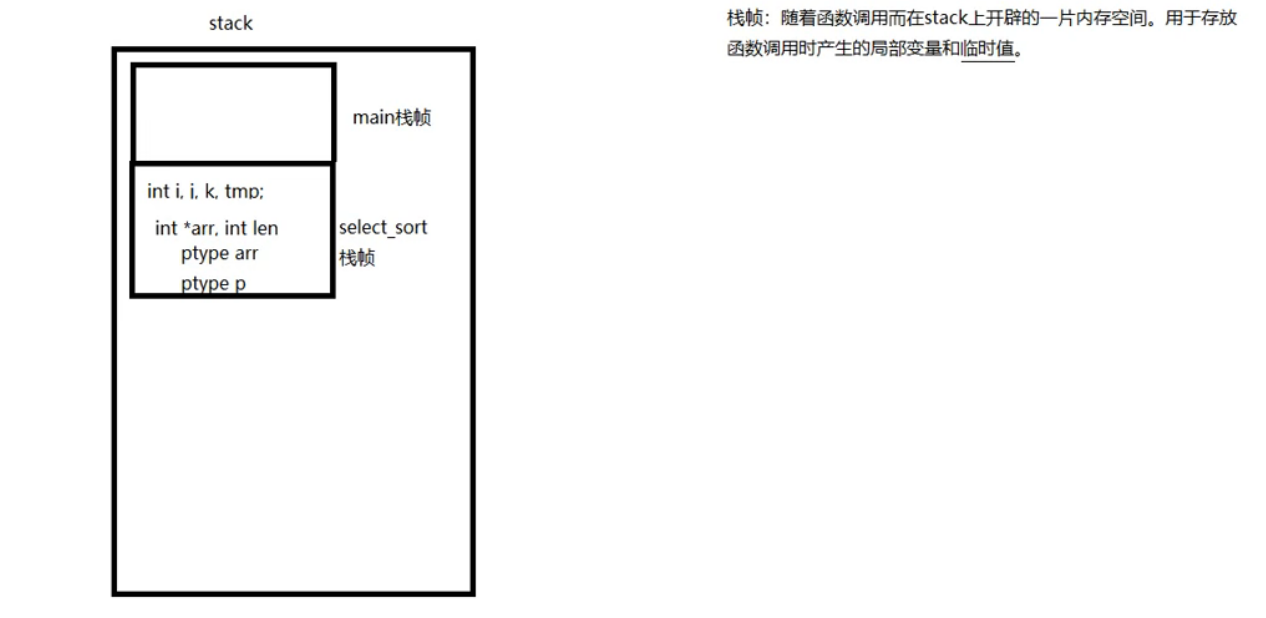
backtrace/bt:列出当前程序正存活的栈帧

frame/f 切换函数的栈帧

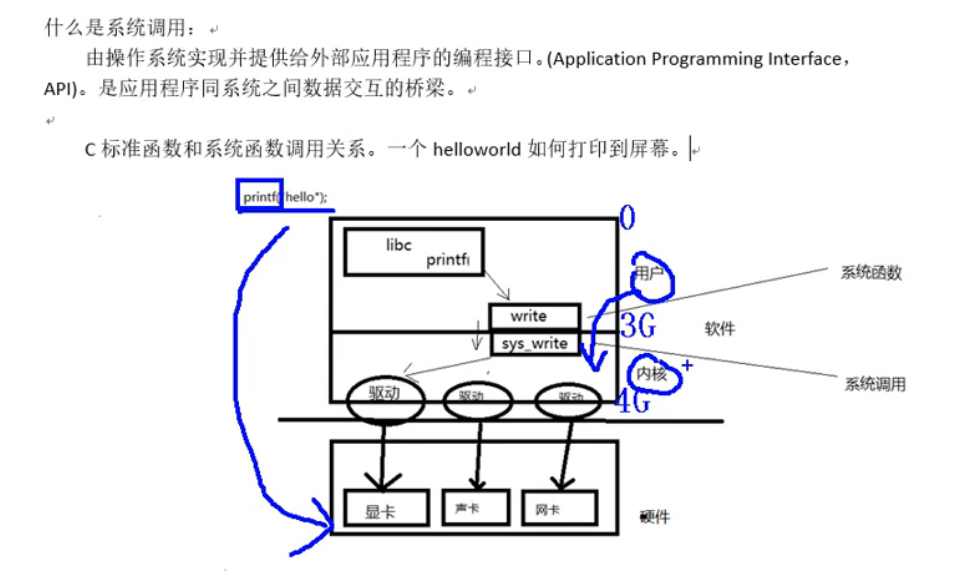
display:设置跟踪变量

undisplay：取消设置跟踪变量，使用跟踪变量的编号

12.



13.



14.文件描述符就是一个整数

open函数：

int open(char \*pathname, int flags) #include<unistd.h>

参数：

pathname:欲打开的文件路径名

flags：文件打开方式，O\_RDONLY|O\_WRONLY|O\_RDWD #include<fcntl.h>

返回值：

成功：返回打开文件所得到对应的文件描述符（整数）

失败：-1，设置errno

int open（char \*pathname, int flags, mode\_t mode）

参数：

pathname:欲打开的文件路径名

flags：文件打开方式，O\_RDONLY|O\_WRONLY|O\_RDWD

mode:参数三使用的前提是，参二指定了O\_CREAT。取值8进制数，用来描述文件的访问权限。rwx 0664

创建文件最终得到的权限= mode & ~umask

返回值：

成功：返回打开文件所得到对应的文件描述符（整数）

失败：-1，设置errno

close函数

int close(int fd)

参数前面函数返回的文件描述符

错误处理函数： 与errno相关

printf（“xxx error： %d\n”,errno）

char \*strerror(int errnum);

printf(“%s\n”,strerror(errnum))

void perror(const char \*s);

perror(“open error”);等于实现了printf(“%s\n”,strerror(errnum))

read函数：

ssize\_t read(int fd, void \*buf, size\_t count);

参数：

fd：文件描述符

buf：存数据的缓冲区

count：缓冲区大小

放回值：

0表示读到文件末尾

成功：读到的字节数

失败：-1，设置errno

-1：并且errno = EAGIN 或EWOULDBLOCK,说明不是read失败，而是read在以非阻塞方式去读一个设备文件（网络文件）并且文件无数据

write函数：

ssize\_t write(int fd, const void \*buf, size\_t count);

参数：

fd：文件描述符

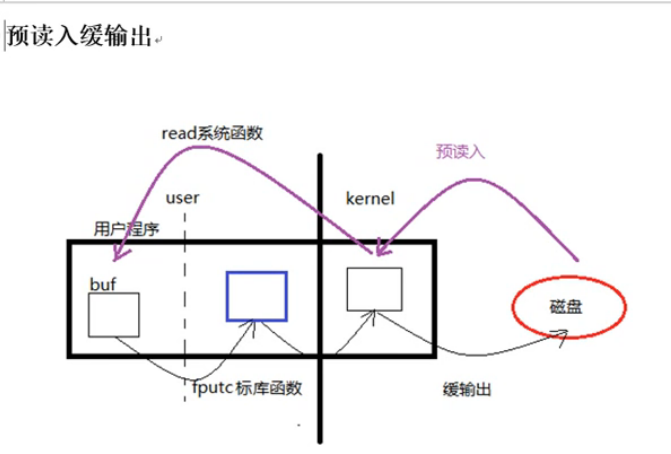
buf：待写出数据的缓冲区

count：数据大小

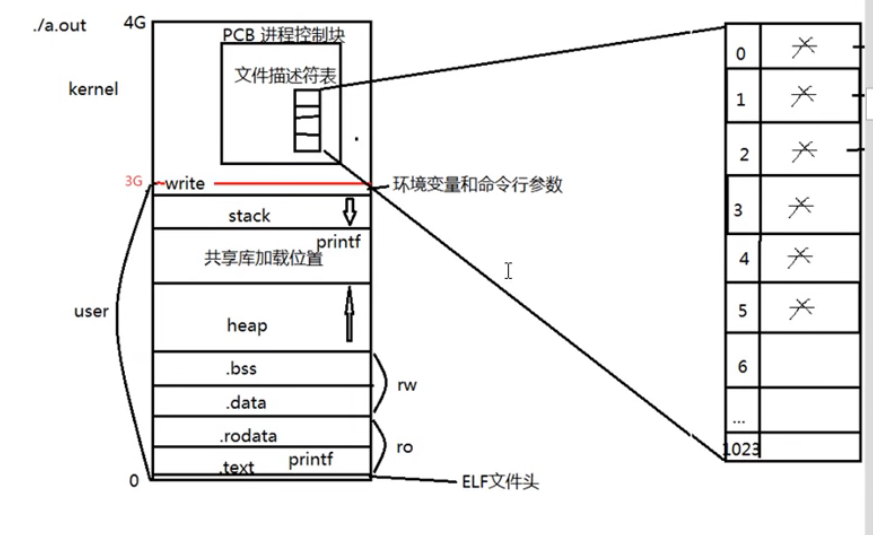
放回值：

成功：写入的字节数

失败：-1，设置errno



文件描述符：



PCB进程控制块，本质结构体

成员，文件描述符表

文件描述符：0/1/2/3/…./1024 表中可用的最小的

0 – STDIN\_FILENO

1 – STDOUT\_FILENO

2 – STDERR\_FILENO

15.

阻塞和非阻塞： 是设备文件，网络文件的属性

产生阻塞的场景。都设备文件/和读网络文件。（都常规文件无阻塞概念）

/dev/tty –终端文件。

open(“/dev/tty”, O\_RDWR|O\_NONBLOCK) --设置/dev/tty非阻塞状态。（默认为阻塞状态）

8.

进程：

程序：死的。只占用磁盘空间。 --剧本

进程：活的。运行起来的程序，占用内存、cpu等系统资源。 --戏