学习目标

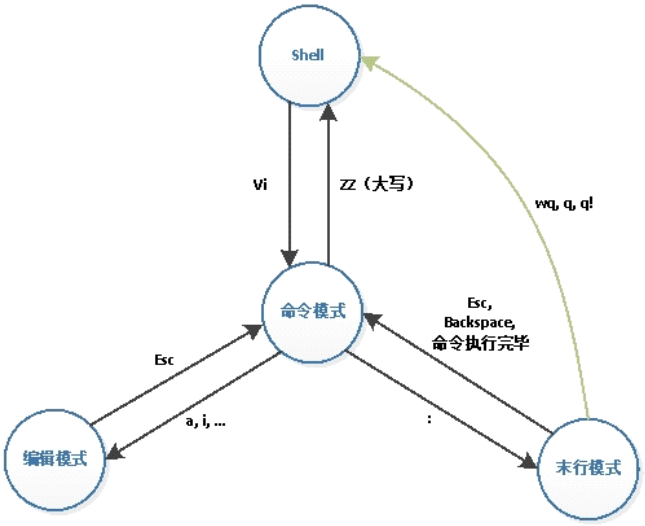
1. 掌握vim命令模式下相关命令的使用
2. 掌握从命令模式切换到编辑模式的相关命令
3. 掌握vim末行模式下相关命令的使用
4. 能够说出gcc的工作流程和掌握常见参数的使用
5. 熟练掌握Linux下的静态库的制作和使用
6. 熟练掌握Linux下的共享库的制作和使用

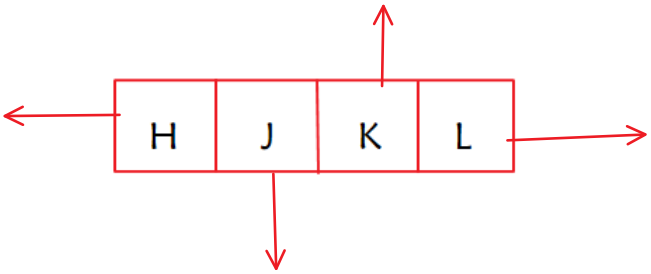
00 - 软件的安装和卸载

1. 在线安装 -- ubuntu apt-get
    - 安装: sudo apt-get（apt）install 安装包的名字
    - 卸载: sudo apt-get（apt）remove 软件的名字
    - 软件列表的更新: sudo apt-get（apt） update
    - 清空缓存: sudo apt-get（apt） clean
        - /var/cache/apt/archives
            - xxx.deb
2. 软件包安装(Ubuntu下 .deb格式)
    - 安装: sudo dpkg -i xxx.deb
    - 卸载: sudo dpkg -r 软件的名字

01 - vim相关

1. 需要自己安装vim
   - shell敲命令：vimtutor
2. vim的三种工作模式
   a. 命令模式
   b. 编辑模式
   c. 末行模式





3. vim命令模式下的相关操作
   - 保存退出: ZZ
   - 代码格式化: gg=G
   - 光标的移动
     - 上下左右:
     - 光标移动到行首: 0（零）
     - .....行尾: $
     - 光标移动到文件首部: gg
     - .....文件尾部: G
     - 行跳转123行: 123G
       - 行号n回车: 当前行向下移动n行
   - 删除命令
     - 删除字符:
       - 光标前的字符：X
       - 。。后。。。：x
     - 单词: dw
       - 删除整个单词光标应该在单词的最前边
     - 删除光标前字符串: d0
     - 删除光标后.....: d$(D)
     - 删除行: dd, 删除光标所在行
     - 删除多行: ndd
       - n行数

- 撤销和反撤销
  - 撤销：u
  - 反撤销：ctrl+r
- 复制和粘贴
  - 复制：
    - 1行：yy
    - 多行：nyy
  - 粘贴：
    - p：粘贴到光标的下边一行
    - P：。。。。。上边一行
- 可视模式：v
  - 移动光标：
    - hjkl
  - 复制：y
  - 删除：d
  - 粘贴：
    - p：粘贴到光标的后边
    - P：粘贴到光标的前边
- 替换操作
  - r：替换一个，光标盖住的字符
  - R：替换多个，从光标盖住的这个往后替换
- 查找命令
  - /xxx
  - ?xxx
    - 关键字切换：n/N
  - #:
    - 光标移动到待搜索关键字上边，键盘输入#
- 查看man文档
  - 章节号+K（shift+k）

4. 命令模式切换到文本编辑模式
   - a：从光标后开始插入字符
   - A：行尾
   - i：光标前插入
   - I：行首
   - o：光标下创建新行
   - O：。。。上
   - s：删除光标盖住的字符
   - S：删除光标所在行
5. vim末行模式下相关操作
   a. 从命令模式 -> 末行模式
      i. 键盘录入冒号（：）

- 保存退出
  - 保存不退出：w
  - 退出：q
  - 退出不保存：q!
  - 保存退出：wq ==x
- 替换
  - 替换光标所在行的字符串:
    - :s/old/new/gc
      - g: 替换当前行所有的old
      - c：替换的时候添加提示信息
  - 替换一个范围
    - :x,ys/old/new/gc
      - xy是一个范围（行数）
  - 替换当前文档所有的
    - :%s/old/new/gc
- 分屏操作
  - 当前文件分屏
    - 水平：sp
    - 垂直：vsp
  - 两个屏幕显示不同的文件
    - 水平：sp 文件名
    - 垂直：vsp 文件名
  - 屏幕的关闭:
    - 关闭所有：qall
    - 保存关闭所有：wqall
    - 保存所有：wall
  - 屏幕的切换:
    - ctrl+w+w
  - 打开的时候分屏
    - 水平：vim -on 文件名 文件名
    - 垂直：vim -On 文件名 文件名
      - n可以省略的，表示分屏的个数
- 执行shell命令
- 行跳转
  - ：行号（回车）
- 末行模式 -> 命令模式:
  - 两次esc
  - 末行模式下执行一个命令

6. vim配置文件
   - 用户级别：
     - ~/.vimrc

- 系统级别：
  - /etc/vim/vimrc

h j k l

02 - gcc相关

1. gcc工作流程
   - 预处理 - -E
     - 宏替换
     - 头文件展开
     - 注释去掉
     - xxx.c -> xxx.i
       - c文件
   - 编译 - -S
     - xxx.i -> xxx.s
     - 汇编文件
   - 汇编 - c
     - xxx.s -> xxx.o
     - 二进制文件
   - 链接
     - xxx.o -> xxx(可执行)
2. gcc常用参数
   - -v/--version
   - -I: 编译的时候指定头文件路径
   - -c:
     - 将汇编文件生成二进制文件，得到了一个.o文件
     - 源代码
   - -o: 指定生成的文件的名字
   - -g:
     gdb调试的时候需要加
   - -D

在编译的时候指定一个宏

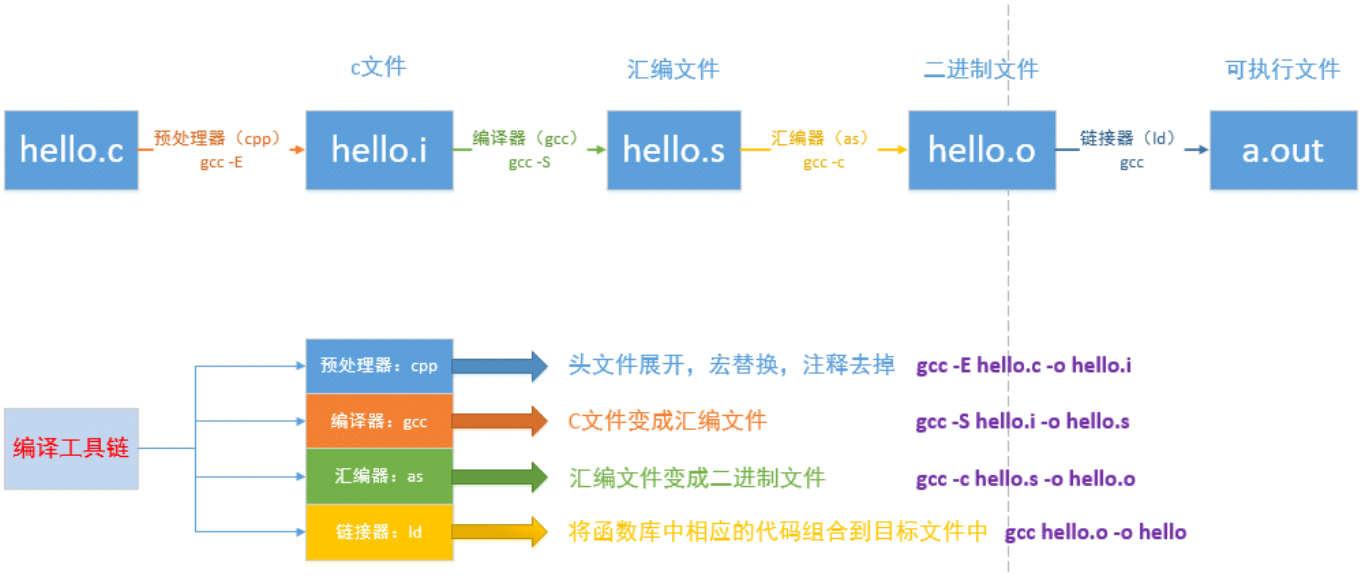
- 使用场景：测试程序的时候用

- -Wall:
  - 添加警告信息
- -On:
  - 优化代码，n是优化级别：1,2,3

```
int a = 10;
int b = a;
int c = b;
printf("%d", c);

int c = 10;
printf("%d", 10);
```

-- gcc工作流程图


c文件
汇编文件
二进制文件
可执行文件
hello.c
预处理器（cpp）
gcc -E
hello.i
编译器（gcc）
gcc -S
hello.s
汇编器（as）
gcc -c
hello.o
链接器（ld）
gcc
a.out
编译工具链
预处理器：cpp
编译器：gcc
汇编器：as
链接器：ld
头文件展开，宏替换，注释去掉
gcc -E hello.c -o hello.i
C文件变成汇编文件
gcc -S hello.i -o hello.s
汇编文件变成二进制文件
gcc -c hello.s -o hello.o
将函数库中相应的代码组合到目标文件中
gcc hello.o -o hello

03 - 静态库和动态库的制作和使用

1. 库是什么?
    - 二进制文件
    - 将源代码 - 》二进制格式的源代码
        - .c .cpp
    - 加密
2. 库制作出来之后,如何给用户使用?
    - 头文件
    - 制作出的库
3. 静态库的制作和使用
    - 命名规则: libtest.a
        - lib
        - xxx - 库的名字
        - .a
    - 制作步骤:
        - 原材料: 源代码.c .cpp
        - 将.c文件生成.o
            - gcc a.c b.c -c
        - 将.o 打包
            - ar rcs 静态库的名字 原材料
            - ar rcs libtest.a a.o b.o
                - ar - archive
    - 库的使用:
        - gcc test.c -I ./ -L./lib -lmycalc -o app
            - -L: 指定库的路径
            - -l: 指定库的名字取得lib和.a
4. 动态库的制作和使用

- 命名规则
  - libxxx.so
- 制作步骤
  - 将源文件生成.o

    gcc a.c b.c -c -fpic(fPIC)
  - 打包

    gcc -shared a.o b.o -o libxxx.so
- 库的使用
  - 头文件a.h
  - 动态库 libtest.so
  - 参考函数声明编程测试程序 main.c

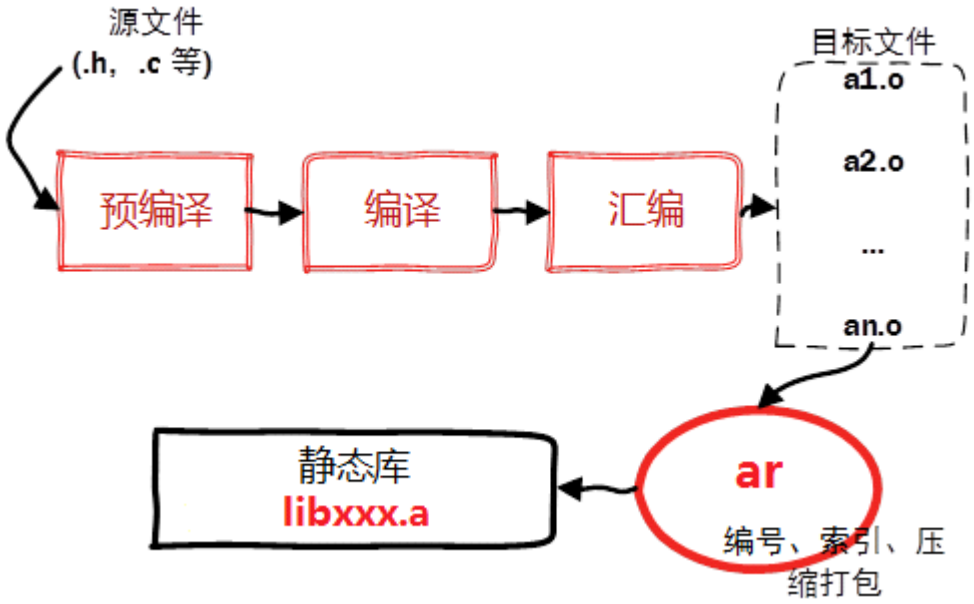
    gcc main.c -I ./ -L ./ -l test -o app
- 动态库无法加载:
  - 使用环境变量
    - 临时设置：

      在终端：

      export LD_LIBRARY_PATH=动态库的路径:$LD_LIBRARY_PATH
    - 永久设置：
      - 用户级别：
        - ~/.bashrc
          - 配置完成：
            - 重启终端
            - source ~/.bashrc
      - 系统级别：
        - /etc/profile
          - source /etc/profile
  - /etc/ld.so.cache文件列表
    - 找到一个配置文件
      - /etc/ld.so.conf

        - 把动态库的绝对路径添加到文件中
    - 执行一个命令：
        - sudo ldconfig -v
- 知识点拓展：
    - dlopen, dlclose, dlsym

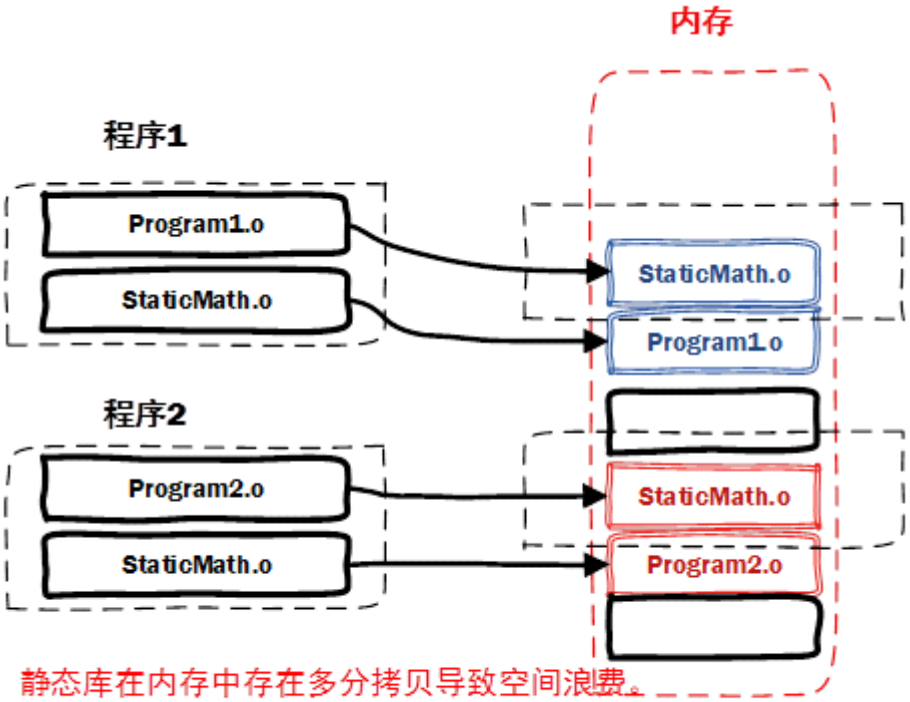
-- 静态库创建过程



-- 静态库，动态库优缺点

静态库：

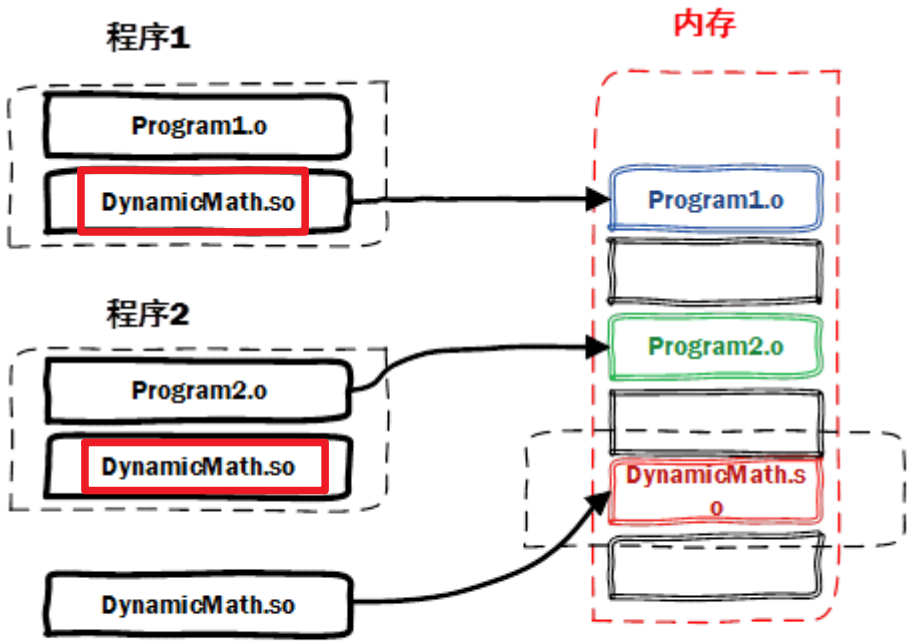
- 优点：
  - 静态库被打包到应用程序中加载速度快
  - 发布程序无需提供静态库，移植方便
- 缺点：
  - 销毁系统资源，浪费内存
  - 更新、部署、发布麻烦。



动态库：

- 优点：
  - 可实现进程间资源共享
  - 程序升级简单
  - 程序猿可以控制何时加载动态库
- 缺点：
  - 加载速度比静态库慢

- 发布程序需要提供依赖的动态库



动态库在内存中只存在一份拷贝，避免了静态库浪费空间的问题。