# Day1:

## C语言：

1. ~是按位取反符
2. 0前缀表示是个八进制数
3. 位运算符

& 与 两个位都为1时，结果才为1

| 或 两个位都为0时，结果才为0

^ 异或 两个位相同为0，相异为1

~ 取反 0变1，1变0

<< 左移 各二进位全部左移若干位，高位丢弃，低位补0

>> 右移 各二进位全部右移若干位，对无符号数，高位补0，有符号数，各编译器处理方法不一样，有的补符号位（算术右移），有的补0（逻辑右移）

二进制数正副判断：

1. 最高位为0是正数 1为负数
2. 枚举：enum Day{mon=1, tue} day;其中Day为枚举名称，day为枚举变量，这是第一种定义方式

也可以 enum {} day; 忽略枚举名称，直接定义枚举变量

其中的mon为枚举元素，枚举元素默认为其对应的位号，可以对其自定义元素值

1. 函数指针：指向函数的指针变量 typedef int (\*fun\_ptr)(int, int);
2. 回调函数：函数指针作为某个函数的参数的函数 void populate\_array(int \*array, size\_t arraySize, int (\*getNextValue)(void))
3. 结构体： struct tag {

member-list

member-list

member-list

} test; // tag是结构体标签, member-list是标准的变量定义，variable-list 是结构体变量，必须得定义结构体变量后才能对结构体成员进行定义

1. ”.” 是成员访问运算符，例如test.me... = 1;

结构体作为函数参数

1. 动态数组定义：

动态定义多维数组：

int \*\*node;

int col,row;

scanf("%d", &col);

scanf("%d", &row);

node = (int \*\*)malloc(col \* sizeof(int \*));

for (int i = 0; i < n - 1; i++)

{

node[i] = malloc(sizeof(int) \* row);

for(int j=0;j<row;j++){

scanf("%d", &node[i][j]);

}

}

动态定义一维数组：

//定义

int n;

int \*num;

scanf("%d", &n);

num = (int \*)malloc(n \* sizeof(int));

//使用

for (i = 0; i < n; i++)

{

scanf("%d", &num[i]);

}

数组初始化：

1. 逗号分隔符
2. 指定初始化int arr[6] = {[3]=0};
3. c99之前不允许int arr[a]; 这样来定义数组
4. 条件运算符 ?:
5. Break, continue 只会影响包含其的循环
6. double 格式化类型输入 必须为 %lf，float 必须为 %f

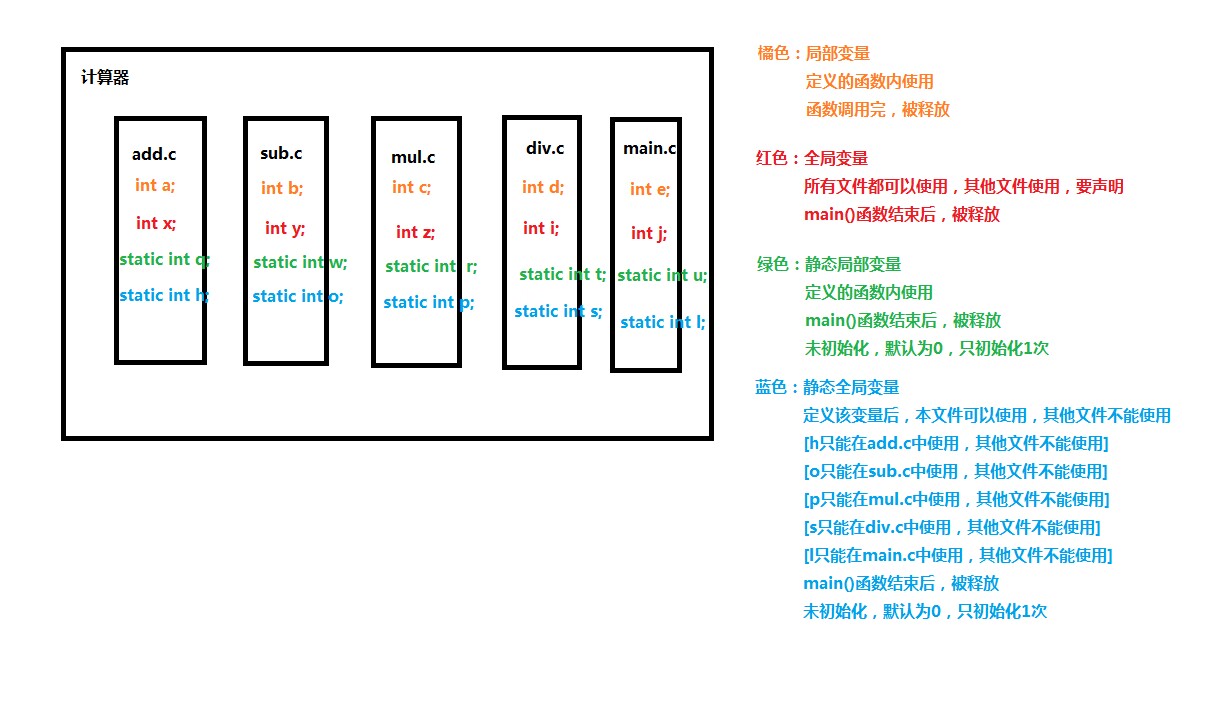
### 指针

1. 函数指针用于通过指针变量调用函数：

int sum(int a, int b);

int (\*fun)(int, int);

1. 如果指针变量是动态的(auto)，则其不会被初始化，而类型为静态(static)的指针变量会将其初始化为0
2. NULL指针表示不指向任何东西
3. 局部变量与全局变量



## vim编辑器使用

1. vi三种模式:命令行模式，插入模式，底行模式
2. 保存当前编辑状态并执行命令:! Command
3. i插入 I行首转插入
4. a字母尾部转插入 A行尾转插入
5. o换行转插入 O上一行转插入
6. 常用命令：

[数量]x 剪切多个字

[数量]yy 复制多行

[数量]dd 剪切多行

u 撤回 ctrl+R 撤回撤回

:q! 强制退出

:w File\_Name 另存为File\_Name并打开

:r File\_Name 快速引入文件(可以快速引入一些自定义头文件)

:N 光标移动到第N行

1G 光标移动到第一行

G 光标移动到最后一行

:set nu 打开vi行号功能

:set nonu 关闭行号功能

/关键字 查找当前文档中的关键字, N向上查找, n向下查找(支持正则表达式, /^关键字 表示匹配除关键字外所有字符)

:范围 s/old/new/替换范围 替换功能 (.表示当前行, 若不标注范围则默认替换当前行的内容，$表示最后一行, %表示全文替换// 替换范围：g表示全文替换)

部分行的操作：

y0 将光标至行首的字符拷入剪贴板

y$ 将光标至行尾的字符拷入剪贴板

d0 删除...

d$

范围 y 块复制

范围 d 块删除

# Linux下常用命令

## 基本常用命令

1. 查询命令使用：man, --help
2. 命令的执行：command [-options] parameter1 parameter2

[]并不存在于实际的命令中，其表示可选， ‘-‘表示选项，’--’表示选项的完整全名，以 ’|’ 分隔的选项不可以一起使用

## apt常用命令

1. apt 语法：apt [options] [command] [package …]
2. apt list --upgradeable 列出可更新的软件包及版本信息
3. sudo apt full-upgrade 升级软件包，升级前先删除需要更新软件包
4. sudo apt show <package name> 显示软件包具体信息，例如版本号，安装大小，依赖关系
5. sudo apt remove <package name> 删除软件包
6. sudo apt auto remove 清理不再使用的依赖和库文件
7. sudo apt purge <package name> 移除软件包及其配置文件

## Bash下常用命令：

Ctrl+u/ctrl+k: 从光标处向前删除/向后删除

Ctrl+a/ctrl+e: 将光标移动到行首/行尾

Ctrl+d: 直接退出终端

1. Bash下环境变量的设置：

变量在被使用时必须在变量前面加上$

变量的使用：

echo $变量名 或 echo ${变量名}

定义变量值：用’=’连接即可，连接处不能有空格，双引号定义的变量值能保留其原本的特性，而用单引号则仅保存为纯文本，可以用转义符’\’来表示一些特殊字符，通常系统变量名为大写，自定义变量名为小写

1. 一些系统变量的含义

HOME 用户根目录

SHELL 告诉我们目前这个环境显示的SHELL是哪个程序

HISTSIZE 系统记录执行过的命令的条数

MAIL 当使用

# Git

1. OpenSSL SSL\_connect: SSL\_ERROR\_SYSCALL in connection to github.com:443

git config --global --unset http.proxy#删除配置

git config --global http.proxy 127.0.0.1:xxxx#根据自己的代理插件设置全局端口

git config --local http.proxy 127.0.0.1:xxxx #为某个 git 项目单独设置代理

1. Git: ‘LF will be replaced by CRLF the next time Git touches it‘

方法一：

#提交时转换为LF，检出时转换为CRLF

git config --global core.autocrlf true

方法二：

#提交时转换为LF，检出时不转换

git config --global core.autocrlf input

方法三：

#提交检出均不转换

git config --global core.autocrlf false

#拒绝提交包含混合换行符的文件

git config --global core.safecrlf true

#允许提交包含混合换行符的文件

git config --global core.safecrlf false

#提交包含混合换行符的文件时给出警告

git config --global core.safecrlf warn