Linux C编程

1. C 语言基础

C语言的设计理念：自顶向下、结构化和模块化设计

程序告诉计算机应该如何完成一个计算任务，程序由一系列指令（Instruction）组成。

key point:

```
#define assert(cond) if(!(cond)) pandic();
```

assert() 是一个宏，不是一个函数，如果出现：

```
if (...) assert(0);
else ...

//上面代码编译器翻译为
if(...)
    if(!0) panic();
else ...                //由于else与最近的if配对，所以会出现错误

/*
*下面是正确的写法
*/
#define assert(cond) \
  do { \
    if (!(cond)) { \
      fprintf(stderr, "Fail @ %s:%d", __FILE__, __LINE__); \
      exit(1); \
    } \
  } while (0)

#define assert(cond) ({ ... })
```

assert**主要用于类型检查及单元测试中**

1.2 C 语言预处理

C语言预处理的步骤为：

- （1）三连字符替换成相应的单字符
- （2）把\字符续行的多行代码拼接成一行

```
#define STR "Hello, "\
                        "world"
//预处理之后
#define STR "Hello, "  "world"
```

这种续行写法要求\后面紧跟换行，中间不能有其他空白字符

- （3）把注释（不管是单行注释还是多行注释）都替换成一个空格
- （4）经过上两步之后去掉了一些换行，有的换行在续行过成中去掉，有的在多行注释之中；剩下的代码成为逻辑代码行。预处理器把逻辑代码行划分为Token和空白字符，这时的Token成为预处理Token，包括标识符、整数常量、浮点型常量、字符常量、字符串、运算符和其他符号
- （5）在Token中识别出预处理指示，做出相应的预处理动作，如果遇到#include预处理指示，则把相应的源文件包含进来，并对源文件做以上四步预处理。遇到宏定义进行宏展开；**在预处理指示中允许使用的空白字符只有空格和Tab**
- （6）找出字符常量或字符串中的转义序列，用相应的字节来替换它，比如把\n替换成字节0x0a
- （7）把相邻的字符串连接起来
- （8）经过以上处理之后，把空白字符丢掉，把Token交给C编辑器做语法分析；此时的Token**成为C Token**

1.2.1 宏定义

较大的项目都会用大量的宏定义来组织代码，宏展开：通过**复制/粘贴**改变代码的形态：#include 粘贴文件、 aa、bb粘贴符号

（1）函数式宏定义

（2）内联函数

C99 引入一个新关键字 **inline**, 用于定义内涵函数（inline function），内联函数在内核代码中很常见，例如include/linux/rwsem.h中：

```
static inline void down_read(struct rw_semaphore *sem)
{
    might_sleep();
    rwsemtrace(sem, "Entering down_read");
    __down_read(sem);
    rwsemtrace(sem, "Leaving down_read");
}
```

inline 关键字告诉编辑器，这个函数的调用要尽可能快，可以当普通的函数调用实现，也可以用宏展开实现。

2. C语言深入理解

2.1 strtok()函数

The C library function **char *strtok(char *str, const char *delim)** breaks string **str** into a series of tokens using the delimiter **delim**.

Following is the declaration for strtok() function.

```
char *strtok(char *str, const char *delim)
```

Parameters

- **str** − The contents of this string are modified and broken into smaller strings (tokens).
- **delim** − This is the C string containing the delimiters. These may vary from one call to another.

Return Value

This function returns a pointer to the first token found in the string. A null pointer is returned if there are no tokens left to retrieve.

2.2 sscanf()函数

2.3 getopt()

program invocation 程序调用

3. Linux C编程

附录

1. Makefile

make 命令执行时，需要Makefile文件，Makefile文件规定了make命令如何编译和链接程序。

make 命令会自动读取当前目录下的Makefile文件，完成相应的编译步骤。Makefile由一组规则组成，每条**Makefile 规则**格式为：

```
target :prerequisites ...
        command1
        command2
        ...
```

target （目标）：目标文件，也可以是执行文件，还可以是一个标签；

prerequisites ：生成target所需要的文件（条件）

command： make需要执行的命令（任意的shell命令）

欲更新目标，必须首先更新它的所有条件，所有条件中只要有一个条件被更新，目标也随之被更新。**更新就是执行一遍规则中的命令列表，命令列表中的每条命令必须用Tab开头，且不能有空格。**

例如：

```
main: main.o stack.o maze.o
    gcc main.o stack.o maze.o
```

//例子讲解

2. Linux常用命令

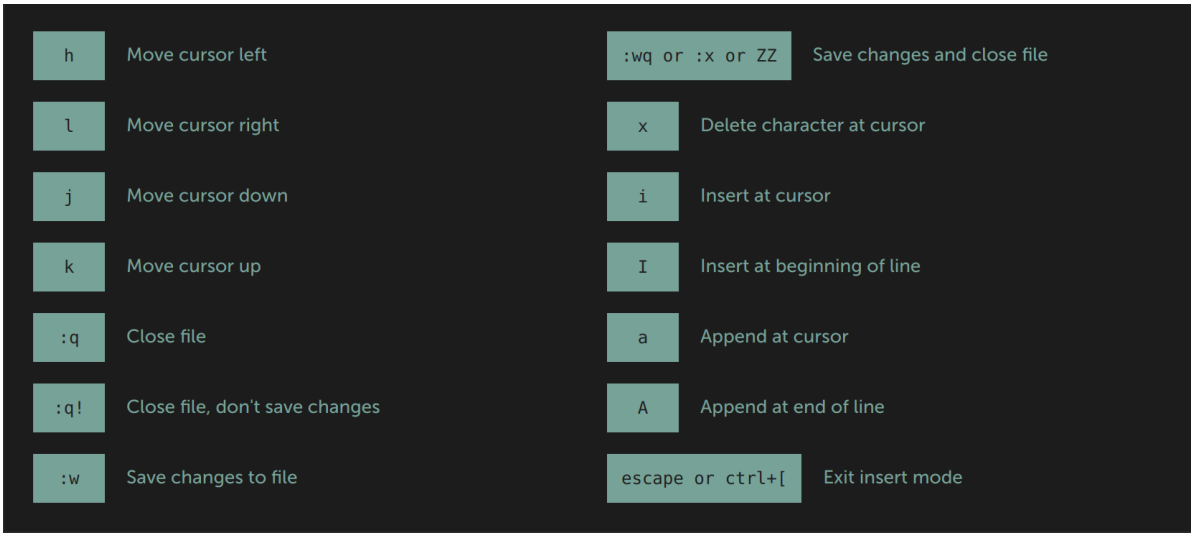
- 文件管理 - `cd`, `pwd`, `mkdir`, `rmdir`, `ls`, `cp`, `rm`, `mv`, `tar`
- 文件检索 - `cat`, `more`, `less`, `head`, `tail`, `file`, `find`
- 输入输出控制 - 重定向, 管道, `tee`, `xargs`
- 文本处理 - `vim`, `grep`, `awk`, `sed`, `sort`, `wc`, `uniq`, `cut`, `tr`
- 正则表达式
- 系统监控 - `jobs`, `ps`, `top`, `kill`, `free`, `dmesg`, `lsof`

```
find  . # 查看当前目录下面所有文件
find . -name "*.c"  # 查看当前目录下所有 .c文件
find . -name "*.c" -o -name "*.h" # 查看当前目录下.c 或者是.h文件 -o: or
find . -name "*.c" -o -name "*.h" | xargs cat   # 管道，将find的结果送入管道作为xargs 的输入
-wc  # word count
vim . # 在终端打开当前目录下所有文件
lscpu # 查看cpu信息
```

- `tree` 名令

```
# tree - list contents of directories in a tree-like format
tree -a  #All files are printed.By default tree does  not  print  hidden files
tree -d  #List directories only.
tree -f  #Prints the full path prefix for each file
tree -C  #Useful to colorize output to a pipe.
```

2.1 vim 操作




```

```

2.2 进程

3. Git Tutorial for me

3.1 Git 常用命令