目录

一、前提概要

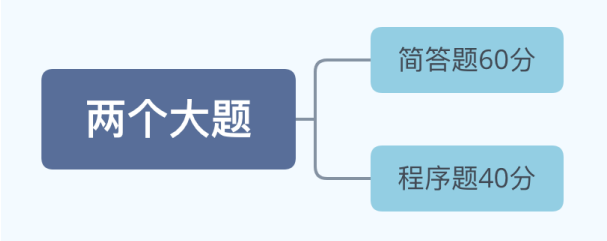

图 1: 考点分布

二、第一章 Linux 快速入门

- **GNU(常识)**

  GNU's Not UNIX，为了推广自由软件的精神以实现的一个自由操作系统。

- **Linux 下目录与分区的关系和 window 的区别 (必须掌握)**

  Window 文件系统是以驱动器的盘符为基础的,并且每一个目录与相应的分区对应。比如 E:\workplace。

  Linux 恰恰相反，文件系统是一个文件树，而且所有的文件和外部设备 (光驱，硬盘) 都是以文件的形式挂载在这颗树上，比如/usr/local。

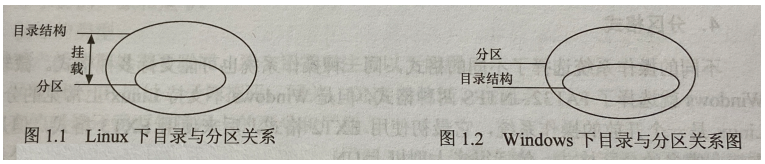

图 2: 文件目录与分区

- **文件属性 (必须掌握)**

  Linux 文件属性你得知道要考, 文件创建者，同组其他用户，其他组的其他用户。

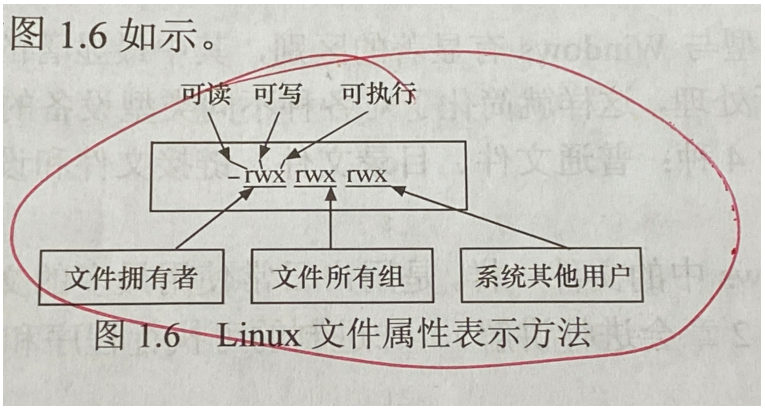

图 3: 属性

可能还是有点抽象，下图为 macbook 下的文件属性

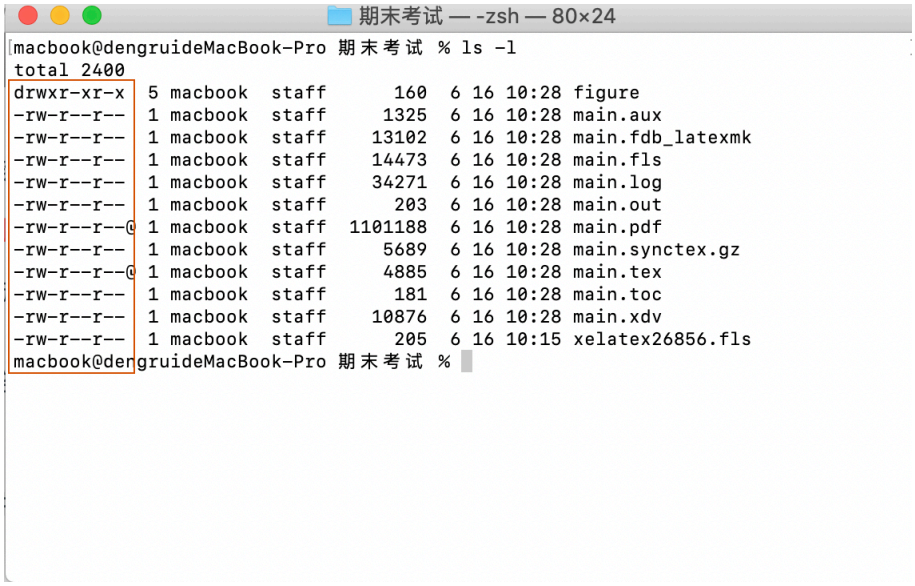

图 4: 文件属性表示

- **文件类型 (知道)**

  1、 “-”表示普通文件

  2、 “d”表示目录文件

  3、 “l”表示链接文件

  4、 “c”表示字符设备

  4、 “b”表示块设备

  5、 “p”表示命名管道

  6、 “f”表示堆栈文件

  6、 “s”表示套接字

三、第二章:Linux 基础命令

- **X 窗口系统 (必须清楚)**

  这个 X 窗口系统实际上就是 Linux 图形化界面 X 窗口系统 (简称 X) 的一部分, 不过你需要注意 **X 窗口仅仅是 Linux 系统上的一个软件 (或者服务), 可以关闭的**, 它不是 Linux 自身的一部分，这一点和 window 不一样，window 窗口系统是嵌入到内核的。

- **Linux 基础命令 (不会直接考)**

  – **su**：切换用户

  用法：su root

  – **useradd**：添加用户账号

  用法：useradd david

  – **passwd**：更改对应用户的账号密码

  用法：passwd david

  – **ps**：显示当前系统中由该用户运行的进程列表。

  用法：ps -ef 查看所有进程及其 PID 号；ps -aux 除了显示-ef 外，还加入 CPU 及内存占用率；pw -w 加宽。

- **cd**：改变当前工作目录
  用法：cd /macbook/奥利给
- **ls**：列出文件和目录信息。
  用法：ls -l 以 long 格式列出文件和目录信息。
- **mkdir**：创建一个目录
  用法：mkdir dengrui
- **cat**：连接并显示一个或者多个文件的有关信息。
  用法：
  cat >abc 将用户输入的值输入到 abc 文件下 (要加回车)，用 control+Z 终止，用 more abc 可以查看。用 » 可以追加内容。
  cat abc > def 将 abc 的内容定向到 def 中，就把 def 内容替换了。
  cat abc » def 将 abc 内容追加到 def 中。
  cat <abc 就是从 abc 读内容了。
- **chomd**：改变文件的访问权限。
  用法：chomd 777 dengrui（-rwx-rwx-rwx：-111-111-111；-rw–rw–rw:-110-110-110）
- **cp**：复制文件
  用法：cp ../我想要复制的文件 ./(当前文件夹下)

四、Linux 下 C 编程基础

- **vi 编辑器 (必须掌握)**：命令行模式、插入模式和底行模式。

  （1）命令行模式

  在用 vi 编辑文件时，最初进入的为一般模式。在改模式下用户可以通过上下移动光标进行“删除字符”或者“整行删除”等操作，也可以进行“复制”、“粘贴”等操作，但是无法编辑文字。

  （2）插入模式

  在此模式下可以进行文字编辑输入，按 [ESC] 键可以回到命令行模式。

  （3）底行模式

  在此模式下，光标位于屏幕的底行。用户可以进行文件的保存或者退出操作，也可以设置编辑环境，如寻找字符串、列出行号等。

- **vi 各模式功能键 (必须掌握)**

  /name：在光标之后查找一个名为 name 的字符串；

  ?name：在光标之前查找一个名为 name 的字符串；

  :set nu：设置行号；

  :set nonu: 取消行号设置；

  yy：复制；

  p：粘贴

- **GCC 编译器 (四个阶段必须知道)**

  1、 预处理

  对包含 (#include) 和 (#define、#ifdef) 等进行处理。编译器将包含的头文件 stdio.h 编译进来，并且用户可以使用 gcc 的选项”-E”进行查看，此选项是让 gcc 在预处理结束后停止编译过程。

gcc -E hello.c -o hello.i。

2、 编译

随后是编译阶段，首先 gcc 要检查代码的规范性、是否有语法错误等，用以确定代码实际要做的工作，检查无误后，gcc 把代码翻译成汇编语言。并且可以用”-S”选项来进行查看，该选项只进行编译不进行汇编，结果生成汇编代码。

gcc -S hello.i -o hello.s

3、 汇编

把编译阶段生成的”.s”文件转成目标文件，在此使用”-c”选项就可以看到汇编代码已转化为”.o”的二进制目标代码了。

gcc -c hello.s -o hello.o

4、 链接

成功编译后进入链接阶段。

重要概念：函数库

在没有特别指定时候，gcc 会到系统默认的搜索路径”/usr/lib”下进行查找，链接到 libc.so.6 函数库中去，这样就能调用函数”printf”了，这就是链接的作用。

函数库有静态库和动态库两种

静态库是指编译链接时，将库文件的代码全部加入可执行文件中，因此生成的文件比较大，但在运行时也就不再需要库文件了。其后缀名通常为”.a”。

动态库与之相反，在编译链接时并没有将库文件的代码加入可执行文件中，而是在程序执行时加载库，这样可以节省系统开销。一般动态库后缀名为”.so”，如前面所述的 libc.so.6 就是动态库。gcc 在编译时默认使用动态库。

- 完成链接后，gcc 就可以生成可执行文件，如下

gcc hello.o -o hello

- 运行该可执行文件：

./hello

- **静态库与动态库区别（重点掌握）**

动态库只有当使用它的程序执行时才被链接使用，而不是将需要的部分直接编译入可执行文件中，并且一个动态库可以被多个程序使用故可称为共享库，而静态库将会整合到程序中，因此在程序执行时不用加载静态库。从而可知，链接到静态库会使用户的程序臃肿，并且难以升级，但是可能会比较容易部署。而连接到动态库会使用户的程序轻便，并且易于升级，但是会难以部署。

- **gdb 调试器（必须掌握）**

gcc -g test.c -o test 生成可执行文件 test；gdb test 对可执行文件进行调式（进入调式页面）

全部细节看书上 68 页

- **make 工程管理器（必须掌握）**

target：dependency_files

command

例如，有两个文件分别为 hello.c 和 hello.h，创建的目标体为 hello.o，执行的命令为 gcc 编译指令：gcc -c hello.c，那么对应的 makefile 就可以写成如下：

Hello.o：hello.c hello.h

gcc -c hello.c -o hello.o

makefile 变量作用: 代词，用更短的字符串代替更长的字符串用来减少代码的书写量。

makefile 规则：通过什么样的方式来编译目标文件。

- **autoTools(不要，不会考)**
- **汉诺塔递归程序**略

五、第六章: 文件 I/O 工程

- 系统调用

  所谓系统调用是指操作系统提供给用户程序调用的一组特殊接口，用户程序可以通过这组特殊接口来获得操作系统内核提供的服务。

- 用户编程接口

  API 实现系统调用途径。

- 系统命令

  系统命令相对于 API 高了一层，实际上是一个可执行程序，他的内部引用了用户编程接口 API 来实现相应的功能。

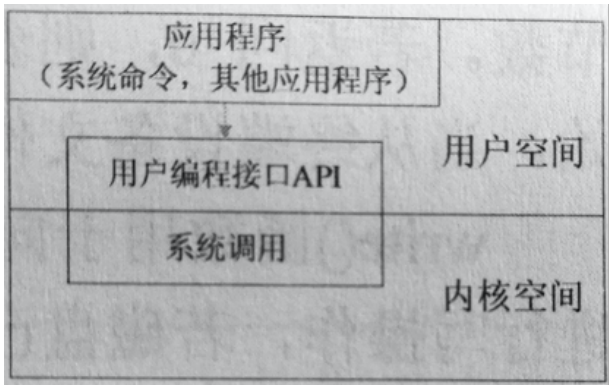

图 5: 系统调用、API 及系统命令之间的关系

- **copy_file.c 书上 P157 页 (重点掌握)**
- **文件锁**

  fcntl( ) 能对文件的某一记录上锁，也就是记录锁。

  记录锁分为读取锁和写入锁，其中读取锁又称为共享锁，它能够使多个进程都能在文件的同一部分建立读取锁。而写入锁又称排斥锁，在任何时刻只能有一个进程在文件的某个部分上建立写入锁。在文件的同一部分不能同时建立读取锁和写入锁。

  上了读锁可以上读锁、上了读锁后不能上写锁、上了写锁后就不能上其他任何锁。

- **multiplex_select.c 书上 P166 页 (熟练掌握)**

  Select 函数来监控哪个终端有字符输入，文件描述符在 FD_ISSET 文件描述符集中

六、第七章: 进程控制开发

- 进程运行的状态

  执行态：该进程正在执行，即进程正在占用 CPU。

  就绪态：进程已经具备执行的一切条件，正在等待分配 CPU 的处理时间片。

  等待态：进程不能使用 CPU，若等待事件发生（等待的资源分配到）则可将其唤醒。

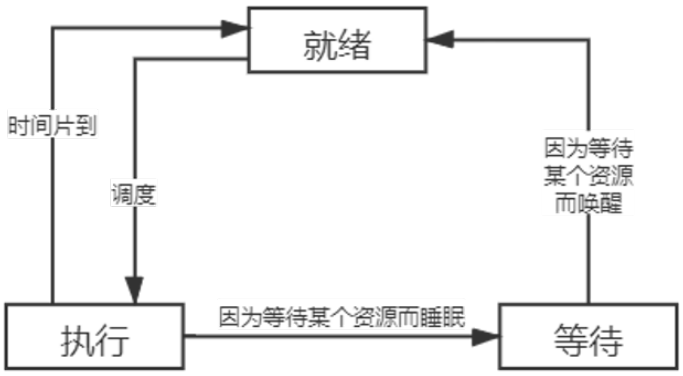

图 6: 进程三种状态的转换关系

- **fork() 函数**

  fork() 用于创建一个子进程，创建的子进程几乎是父进程的完全复制，唯一一个有两个返回值的函数。它在父进程中的返回值是子进程的进程号，在子进程中返回 0。

- **Exec 函数族**

  exec 函数组提供了一个在进程中启动另一个程序执行的方法（使 fork() 创建的子进程完成既定的动作）。

- **exit() 和_exit()**

  两者都用于终止进程，区别在于 exit() 带缓冲而_exit() 不带缓冲。若想保证数据完整则最好使用 exit()。

- **wait() 和 waitpid(), 调试视频认真看**

  两者都用于阻塞主进程，等待子进程退出。子进程未退出，父进程不能先退出。区别在于 waitpid() 可以使用参数 WNOHANG 使父进程不会阻塞。

- **守护进程**

  守护进程就是后台服务进程，它是一个生存期较长的进程，通常独立于控制终端并且周期性地执行某种任务或等待处理某些发生的事件。守护进程常常在系统引导载入时启动，在系统关闭时终止。

  - 创建子进程，父进程退出
  - 在子进程创建新会话
  - 改变当前目录为根目录
  - 重设文件权限掩码
  - 关闭文件描述符

七、第八章: 进程间通讯

- **无名管道**：具有亲缘关系的进程间通信的手段

  创建步骤：在主进程调用 pipe() 函数创建管道，再使用 fork() 函数创建子进程，子进程继承父进程的管道；将父子进程相应的读写端关闭，从而形成父子进程间的一个半双工的通信通道；形成通道以后再调用底层 I/O 函数 read() 和 write() 向这个管道当中一端写另一端读，从而形成父子进程间通信的效果。

- **标准流管道**：固定读写端的管道，写端固定于键盘，读端固定于显示器。

  它将创建无名管道的步骤封装到一个 popen() 函数内直接调用。

  * 注意区分 popen() 函数中的传入参数"r"和"w": "r"产生输出，"w"产生输入。（具体语法及代码见书 P238）

- **有名管道**

  本质是任意两个进程对一个管道文件进行读写操作，一个进程往管道文件里写，另一个进程往管道文件里读，从而形成两个进程间通信的效果。

  注意区分管道文件和普通文件：它们都是文件，在磁盘上有实体文件名存在（不同于无名管道，无名管道只存在与内核空间当中，在文件系统中看不到）；管道文件只是一个运输信息的数据通道，不存字符，不占用字节空间。

- **信号**

  信号是在软件层次上对中断机制的一种模拟，是一种异步通信方式，可以实现用户空间进程和内核进程之间自由切换。

  2 种信号处理方法（了解书上代码）：1、signal() 函数/sigaction() 函数 [P248]；2、信号集函数组 [P251]

- **信号量（重点考察）**信号量是用来解决进程之间的同步与互斥问题的一种进程之间通信机制，包括一个称为信号量的变量和在该信号量下等待资源的进程等待队列，以及对信号量进行的两个原子操作（PV 操作）。信号量相当于一个临界资源，书中信号量为二维信号量，其值非 0 即 1；当它等于 0 时只能进行 V 操作，当它等于 1 时只能进行 P 操作。

  要点：信号量初值；P 操作（-1 操作）；V 操作（+1 操作）；信号量的应用（重点掌握 [P259]

- **共享内存**

  共享内存是一种最为高效的进程间通信方式，因为进程可以直接读写内存而不需要进行数据的复制。

  书上示例代码（了解，能与运行结果对应起来)[P262-265]

- **消息队列**

  消息队列就是一些消息的列表，具有先进先出 (FIFO) 的特征。

  消息队列的实现包括：创建和打开消息队列；添加消息；读取消息；控制消息队列。

  书上示例代码（了解）[P269-271]

- **共享内存实验（认真看）**

  在共享内存实验中利用信号量来约束读写操作（信号量的应用）

八、第九章: 多线程编程

三个重点函数:

Phread_create(): 创建线程

Phread_exit(): 退出线程

Phread_join(): 可以将当前线程挂起来等待线程的结束。

线程有序执行的机制: 互斥锁和信号量，示例代码熟练掌握。

8.1 信号量-生产者消费者实验 P297

九、第十章: 嵌入式 Linux 网络编程

P313 的功能作用要熟悉

基础编程和高级编程能区分的开就可以了，了解代码。