王建立补充：

学习C语言的三句箴言：

1.区分常量和变量

2.先找到变量的地址，然后根据类型访问变量的内容

3.运算符的优先级及其结合性。 例：int (\*p)[3];p是指针类型的变量。

CPU能看到4G内存地址空间(因为系统32位)

CPU构成：运算器，寄存器，控制器

API: Application Programming Interface(应用程序编程接口)

API和系统调用都是系统级函数

APP: Application 应用

HW: Hardware 硬件

OS: Operating System操作系统(的内核)

————————

2.GNU项目

GNU项目组

开放源代码，《GNU宣言》。

GNU项目组开发了很多实用的小项目。

gcc编译器、vi编辑器、bash(是shell其中的一种)

GNU/LINUX操作系是由linux内核和实用的app组成

3.计算机语言的发展历史

开关 灯 状态

1 1 2

2 2 4

3 3 8

。。。 。。。 。。。

32 32 4G

假定的实现操作的机器语言，对应上定义的简便的操作符

11111111，11111111，00000011，00000010 add

11111111，11111110，00000011，00000010 sub

if...else,while,for等分别对应汇编的一个框架。

————————

补充：

1）shell是用户和操作系统交互的窗口用户通过它，对操作系统下达命令。

2）system call(系统调用)本质就是C语言写成的函数。

1. 框架本质是类库
2. start是程序入口,main函数是C语言入口。
3. 专业术语 分层、透明(可以调用但看不见)
4. 显示进程映射

\*\*\*shell命令是可以在shell界面使用的命令(包括当前目录下的命令)

介绍几个shell命令：

1. 使用命令ps -aux找到进程的进程号是4852
2. pstree命令：查看进程树
3. ps -o (pid,ppid,......)：输出指定的字段
4. ps -aux
5. cat /proc/4852/maps
6. cd -：撤销上一次cd命令
7. od -tx1 -tc aaa.txt

可以查看文件的内容，显示十六进制ASCLL码

8.which cat/ps/... 搜索命令(或可执行文件)所在位置

ps命令详解：

ps命令的执行结果如下：

PID - 进程号(重点)

TTY - 终端的编号

TIME - 消耗CPU的时间

CMD - 进程启动的命令名称及参数选项(重点)

------------------------

ps -aux 表示显示所有包含其他使用者的进程

ps -aux | more 表示将ps -aux的结果进行分屏显示

a - 显示所有用户拥有控制终端的进程信息

u - 也包括没有控制终端的进程

x - 以详尽方式显示

w - 以更大列宽显示

ps -aux执行的结果如下：

USER - 进程的属主(熟悉)

PID - 进程号(掌握)

%CPU - 占用CPU的百分比

%MEM - 占用内存的百分比

VSZ - 虚拟内存的大小

RSS - 物理内存的大小

TTY - 终端编号

STAT - 进程的状态信息(熟悉)

START- 进程的启动时间

TIME - 消耗CPU的时间

CMD - 命令的路径和名称以及参数(掌握)

常见的进程状态STAT：

S 休眠状态

s 进程的领导者(拥有子进程)

Z 僵尸进程

R 正在运行的进程

O 可运行的进程

T 挂起状态

< 优先级高

N 优先级低

...

----------------------------

ps -efl 表示以全格式的方式显示当前所有的进程

ps -efl | more 分屏显示命令的执行结果

ps -efl执行结果如下：

PPID - 父进程的进程号

————————————————

补充：

1）echo(1)

echo 字符串：在显示器上显示字符串

echo hello > a.pipe => 写入数据到管道文件中(重定向)

cat a.pipe => 读取管道中的数据

$变量名：取变量的值

echo $变量的名字：将变量的内容显示在屏幕上

echo $?：将上一个命令的执行结果显示到显示器上

在linux上写c程序，规则，程序正常运行结束返回0，

不正常返回非0；

2）man帮助

man 1 command 查看linux命令的帮助信息

man 2 system call 查看系统调用的帮助信息

man 3 标准C库函数名 查看标准C库中的函数的帮助信息

man帮助最后都有使用函数的例子

3）ldd 可执行文件 查看可执行文件依赖的库文件

4）nm命令的使用

nm 二进制文件 //察看文件的导出符号 (例如: nm tmath.o)

T表示： 外部函数，可以在外部调用，有函数实现的代码

U表示： 没有函数实现的代码

————————

静态链接：发生在编译阶段(把函数代码完全拷贝到可执行文件中,变成T)，随处可用

动态链接：程序运行的时候，才发生的链接。(printf)，使用时需要链接库文件

大家都有的函数，就不链接(拷贝)了，节省空间

框架和组件??printf就是一个组件

————————

bash的外部和内部命令

外部命令：bash先fork出来一个进程，再用exec\*加载的一个可执行文件(cp,mv,ls,cat)

内部命令：bash程序中写的函数(例如：cd,pwd)

type command 用于察看命令的内部还是外部。

————————

补充：

制作跟系统一样的库文件

第一步：将头文件移动到系统指定的路径下

/usr/include

(sudo mv 所有头文件 /usr/include)

第二步：将库文件移动到程序链接的时候，默认的路径下

/usr/lib

(sudo mv libtmath.a /usr/lib )

第三步：将程序和库文件链接

gcc test.c -ltmath -o tt

————————

补充：

/usr/lib和/lib是链接器查找库文件的默认路径，同时也是加载器查找库文件的默认路径

客户端更新本质是动态库文件更新

使用新库文件替换旧的库文件，可执行文件不做更改。

第一步：将tmath.c文件中的add函数内部，使用return x-y；

第二步：制作库文件libtmath.so

gcc -fpic -shared tmath.c process.c -o libtmath.so

第三步：更新库文件。

sudo mv libtmath.so /usr/lib

————————

补充：

/etc/passwd 存储了用户的用户名和密码等信息

tarena:x:1000:1000:tarena,,,:/home/tarena:/bin/bash

用户名:密码:用户ID:用户组ID:注释:主目录:shell类型

/etc/shadow 存储了用户的密码信息等

stdin 标准输入 键盘

stdout 标准输出 显示器

stderr 标准错误输出 显示器

————————

补充：??

函数的栈帧与黑客 day04 am 1.55.00

————————————————————————————————

补充：

两种文件的描述符：

0 stdin

1 stdout

2 stderr

0 STDIN\_FILENO 标准输入

1 STDOUT\_FILENO 标准输出

2 STDERR\_FILENO 标准错误输出

linux下一切皆文件，设备和目录也是文件。

在函数的原型中出现...代表这个函数是可变参数。

------------------------------------------------------------------

缓冲文件和非缓冲文件:(标C写的文件，系统调用写的文件)

FILE \*; fopen函数会开辟缓存区，存放将要写入的文件

int fd;

------------------------------------------------------------------

文件流的重定向

那么什么时数据流重定向呢？从字面上理解就是改变数据流的流向，使之流向指定的文件或设备。例如，把执行命令所回传的正确信息（标准输出信息）流向一个文件，而将所回传的错误信息（标准错误输出）流向别一个文件，并把这两个文件的信息保存下来，而在屏幕上不出现任何该命令执行所回传的信息。

通过复制文件描述符，可以改变文件流的流向。

输出重定向与复制的区别

cat open.c >ss 该命令实现输出重定向

额外：《鸟哥私房菜》关于重定向的使用

————————————————————

扩展：

ls -l a.txt详细介绍:

- rwx rwx r-x 1 tarena tarena 0 5月 4 14:49 a.txt

文件类型 属主 属组 其它 硬链接数 属主名称 属组名称 大小 最后修改时间 文件名

-------------------------------------------------------------

在一个文件里，有文件的内容和文件的属性。

文件的元数据，又叫文件的属性，包括大小，权限，创建时间，

-rw-rw-r-- 1 tarena tarena 12 10月 11 10:18 hello

文件的权限里第一字符表示文件类型：

-代表普通文件

d代表文件夹文件

c代表设备文件

b代表块设备文件

l代表符号文件（软链接文件，相当于快捷方式）

s socket文件

p管道文件

硬链接数：对应文件存储的接口数，硬链接数为0的时候文件可能被系统删除。

指向同一个inode，为硬链接，inode索引编号相同

inode为文件存储时的索引节点

补充：

ln(1)命令为文件hello建立硬链接文件h：ln hello h

h和hello两个文件的inode是同一个

使用stat(1)h显示inode信息：stat h

软链接

为h文件建立软链接文件c

ln -s h c

ls -l c:

lrwxrwxrwx 1 tarena tarena 1 10月 11 14:31 c -> h

-------------------------

rw-rw-r--

u g o

a

chmod(1)命令改变文件权限：

chmod u+x 文件名

chmod a+x

chmod u-x

rw-rw-r--

6 6 4

0664 (属主可读可写,属组用户可读,其他可读)

使用按位或的原因：

ls -l a.txt

文件： 读 4 100

写 2 010 |

110 -> 6 可读可写

-------------------------------------------------------------

注意：

文件有权限掩码umask

mode & ~umask

0002

002 -> 自动掩盖其他用户的写权限

umask 命令用来察看umask掩码

umask 权限数字 改变umask掩码的值

------------------------------------------------------------------

linux 下的所有应用进程，是以树状组织的

pstree命令：查看进程树

所有进程之间是以父子或者兄弟组织的

进程0是系统内部的进程，init进程的pid是1，是进程树的根

每当执行a.out程序的时候，bash就fork一个进程出来

假如父进程先结束，子进程变成孤儿进程。子进程的父进程改为init进程(进程号为1)。

————————————————————————————————

补充函数耦合性：

耦合性 耦合性越弱越好，依赖别的函数

内聚性 内聚性越强越好，不依赖别的函数

————————————————————————————————

可重入函数

举例 代码参见reenterable.c

可重入函数：函数中只允许出现局部变量，所有变量的空间都分配在栈帧(栈)里。??

临界资源：有多个执行程序异步访问某个资源的时候，这个资源称为临界资源。

\*\*\*\*进程是资源分配的基本单位，不是程序执行的基本单位。

线程是执行的基本单位

————————————

补充： day10

什么是作业?同一个作业?

作业可以包含多个进程

bash和vi是一个作业，vi的子进程和bash不是一个作业

当发送一个信号时，是给前台作业中所有进程发送

1. 前台作业、后台作业

Ctrl+z 将前台进程切换到后台，处于停止状态

jobs 查看后台的作业号

bg 作业号 在后台运行进程

fg 作业号 将后台进程调到前台来运行

a.out & 直接在后台运行a.out程序，并显示作业号和进程号

所以：ctrl+c 只能给前台进程发送信号。

————————————

网络三层数据交换？？

————————————

#if 0/1

块注释

#endif

函数说明

/\*

function:

para:

return value:

author:

date:

\*/

————————————————————————————————

6.typedef的使用

typedef是给类型起别名。

第一步：先定义变量

int count;

第二步：

typedef int count;

第三步：

count i;

int array\_t[3];

typedef int array\_t[3];

array\_t a; <==> int a[3];

struct node{

int data;

struct node \*next;

};

typedef struct node node\_t;

typedef node\_t \*node\_p;

node\_p p;

node\_t \*\*q; <==> node\_p \*q;

补充：

a)什么是数据类型？

\* 是数据类型。

数据类型就是数据在内存中的访问方式。

指针不是地址，地址属于指针。

int char short float double void \*

int main(int argc,char \*argv[])

int \*p[3];指针类型的数组;数组p有三个元素，每个元素中存放的都是地址。

int (\*p)[3];p是指针类型的变量;指向有三个元素的数组

int (\*fun)(int,int);函数指针

int \*fun(int,int);函数声明