**今天的工作**

熟悉使用git，读代码，了解流程，理框架

**遇到的问题**

~~关于管道，文件重定位~~

~~getuid和geteuid有什么区别？~~

~~getpwuid怎么用~~

**解决的问题**

1. shell是命令解释器
2. git使用
3. git clone 远程仓库地址 将远程仓库克隆到本地仓库
4. git add . 将所有修改了的代码添加到暂存区
5. git commit -m “note”将暂存区的内容放入缓存区准备提交到远程仓库
6. git push将缓存区的内容更新到远程仓库
7. git pull将远程仓库同步到本地仓库（可用git fetch,get merge代替更加安全）
8. git fetch 不会自动merge
9. getuid返回当前进程的实际用户组ID；geteuid返回当前用户的有效用户组ID。

linux系统中每个进程有两个用户id:

①用户ID(uid):说明进程的创建者是谁

②有效用户ID(euid):说明进程对于文件和资源的访问权限（具备哪个用户的权限）

1. getpwuid()
   1. 头文件：#include <pwd.h>

#include <sys/types.h>

* 1. 功能：根据用户uid查找用户的passwd数据（passwd结构在pwd.h中定义了）

struct passwd {

char \*pw\_name; /\*user name \*/

char \*pw\_passwd; /\*user password \*/

uid\_t pw\_uid; /\*user id \*/

gid\_t pw\_gid; /\*group id \*/

char \*pw\_gecos; /\*user real name \*/

char \*pw\_dir; /\*home directory \*/

char \*pw\_shell; /\*shell program \*/

};

* 1. pwd->pw\_name得到username

1. 文件描述符：一个索引，通过索引找到对应文件。

例如当进程需要从标准输入读取数据时就会通过0索引找到标准输入对应的内存缓冲区

* 1. 标准输入——0
  2. 标准输出——1
  3. 标准错误——2

1. 基本流程：
2. 监听用户输入命令并读取
3. 对用户命令进行解析
   1. 内建命令（我们自己编写）
   2. 外部命令（系统bin目录里，在子进程代码中调用exec族函数）——经由shell分析通过linux内核创建新的子进程，在新的子进程中执行。（在新的进程中所执行的代码是不属于 shell 的，所以在 shell 加载时并不随之一起被加载到内存中，而是在外部命令执行时才将其调入内存中。）
4. **不断循环上面的步骤？**
5. 什么是管道（引用https://www.tomorrow.wiki/archives/956）

管道：进程与进程之间通信的通道，例如可以利用管道连接两条命令ls | grep main.c

* + 1. 创建管道 pipe() 需要包含头文件unistd.h

int fd[2];

pipe (fd); //fd[0]读端文件描述符，fd[1]写端文件描述符

* + 1. 创建子进程（此例中父进程为ls，子进程为grep），子进程会继承管道，于是父子进程操作同一个管道
    2. 如ls | grep main.c就是将ls的标准输出写到管道写端，grep的标准输入从管道读端读取

1. 标准输入从管道读端读取

利用dup2(fd[0],0)将fd[0]指向的文件表项赋给文件描述符0即进程默认的标准输入。当进程需要从标准输入读取数据时，进程就会通过文件描述符 0 来找到管道读端所对应内存缓冲区。

close()关闭管道原有的文件描述符而不是关闭管道

1. 什么是文件重定向

和上面的管道差不多原理，只是把标准流文件重定位到指定路径

使用freopen函数(头文件stdio.h):

freopen(const char \*path,const char \*mode,FILE \*stream )

-path:重定位的目的文件

-mode:文件打开的模式-r -w

-stream:标准流文件如stdin,stdout,stderr

-path:重定位的目的文件

-mode:文件打开的模式-r -w

-stream:标准流文件如stdin,stdout,stderr

例如：

//实现了将该进程的标准输出重定向到了文件 out.txt

char fileName[20]="out.txt";

freopen(fileName,"w",stdout);//redirect stdout to fileName

**关键代码**

**1.getHostname**

#include <stdio.h>

#define max\_name\_len 40

/\*\*\*\*\* Info \*\*\*\*\*/

/\* Author: zhj \*/

/\* Function: get hostname\*/

/\*\*\*\*\* Info \*\*\*\*\*/

void getHostname\_zhj(){

const int max\_name\_len = 256;

extern struct passwd \*pwd;

int uid;

char hostname[max\_name\_len];

gethostname(hostname,max\_name\_len);

uid=geteuid();

printf("%s@%d\n",hostname,uid);

}

int main()

{

getHostname\_zhj();

return 0;

}

**2.getUsername**

/\*\*\*\*\* Info \*\*\*\*\*/

/\* Author: zhj \*/

/\* Function: 得到用户名\*/

/\*\*\*\*\* Info \*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#include <pwd.h>

void getUsername\_zhj(){

/\* Note： max\_name\_len是用户名最大的长度\*/

/\*这里定义为40\*/

const max\_name\_len=40;

char username[max\_name\_len];

struct passwd\* pwd = getpwuid(getuid());

strcpy(username, pwd->pw\_name);

printf("%s\n",username);

}

int main()

{

getUsername\_zhj();

return 0;

}

**其他想法**