**OS实验四说明文档**

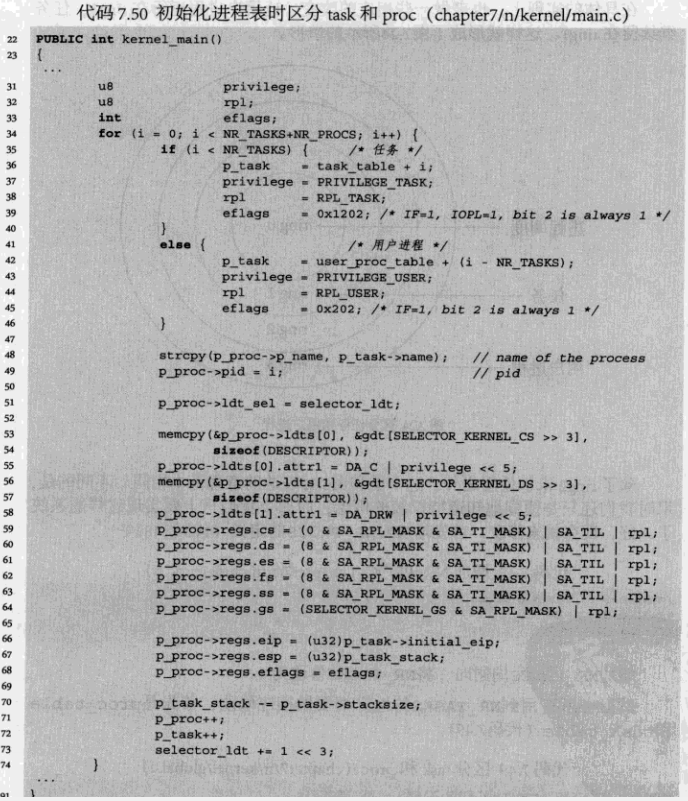
**161250098 彭俊杰**

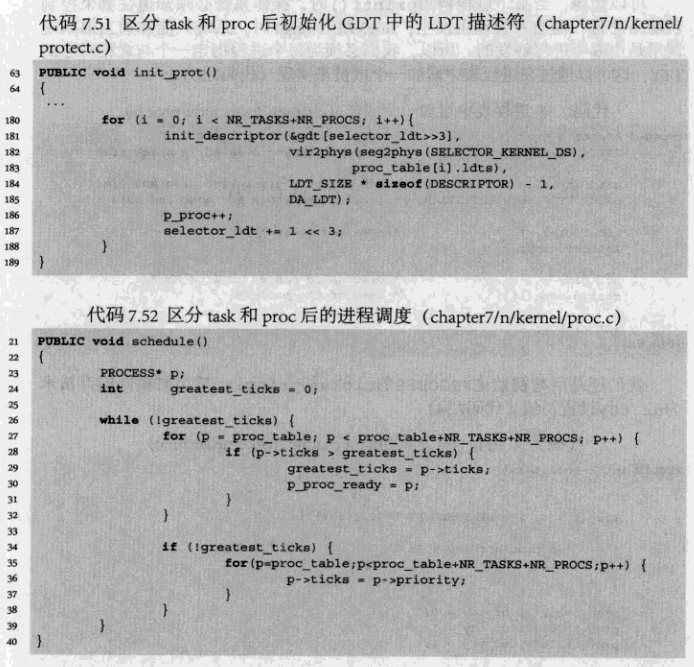
本次实验是在实验三代码的基础上完成的，以下对代码添加及修改部分进行说明：

1. 区分任务和用户进程：

参照《Orange’s》一书7.4节对实验三代码进行修改：

![C:\Users\Kevin\AppData\Roaming\Tencent\Users\1269897230\QQ\WinTemp\RichOle\](7PVD@4H8ML3[F`1UKR{`S.png](data:image/png;base64,)





2、添加A、B、C、D、E五个进程并指定tty：

首先是在kernel/main.c中添加进程体：

void A(){

int i=0;

while(1){

//tty\_table[p\_proc\_ready->nr\_tty].p\_console->char\_color=WHITE;

//my\_disp\_str("A");

//milli\_delay(200);

}

}

void B(){

int i=0x1000;

while(1){

sem\_p(&customers);

sem\_p(&mutex);

waiting--;

sem\_v(&barbers);

sem\_v(&mutex);

tty\_table[p\_proc\_ready->nr\_tty].p\_console->char\_color=BRIGHT\_BLUE;

cuthair(cur\_id,cur\_color);

}

}

void C(){

int i=0x2000;

int index;

while(1){

unsigned int color=GREEN;

sem\_p(&mutex\_i);

index=id++;

sem\_v(&mutex\_i);

sem\_p(&mutex);

if(waiting<CHAIRS){

waiting++;

come\_and\_wait(index,color);

sem\_v(&customers);

sem\_v(&mutex);

sem\_p(&barbers);

cur\_id=index;

cur\_color=color;

}

else{

sem\_v(&mutex);

leave(index,color);

}

process\_sleep(1500);

}

}

void D(){

int i=0x3000;

int index;

while(1){

unsigned int color=YELLOW;

sem\_p(&mutex\_i);

index=id++;

sem\_v(&mutex\_i);

sem\_p(&mutex);

if(waiting<CHAIRS){

waiting++;

come\_and\_wait(index,color);

sem\_v(&customers);

sem\_v(&mutex);

sem\_p(&barbers);

cur\_id=index;

cur\_color=color;

}

else{

sem\_v(&mutex);

leave(index,color);

}

process\_sleep(1500);

}

}

void E(){

int i=0x4000;

int index;

while(1){

unsigned int color=RED;

sem\_p(&mutex\_i);

index=id++;

sem\_v(&mutex\_i);

sem\_p(&mutex);

if(waiting<CHAIRS){

waiting++;

come\_and\_wait(index,color);

sem\_v(&customers);

sem\_v(&mutex);

sem\_p(&barbers);

cur\_id=index;

cur\_color=color;

}

else{

sem\_v(&mutex);

leave(index,color);

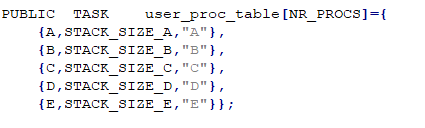
}

process\_sleep(1500);

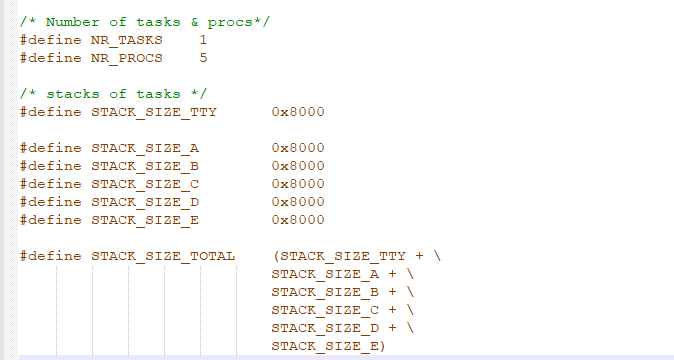
}

}

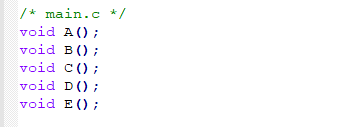
然后在kernel/global.c的user\_proc\_table中添加进程项：



接着修改include/proc.h中的宏定义:

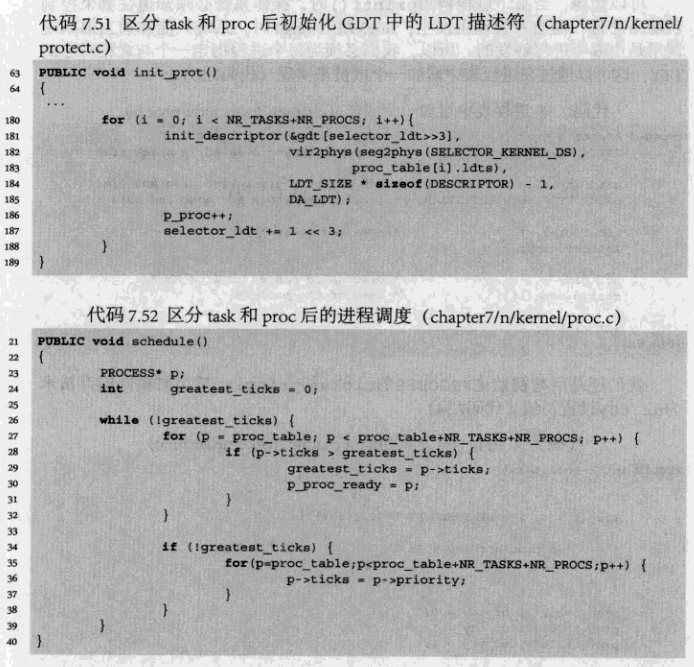


最后在proto.h中添加函数声明：



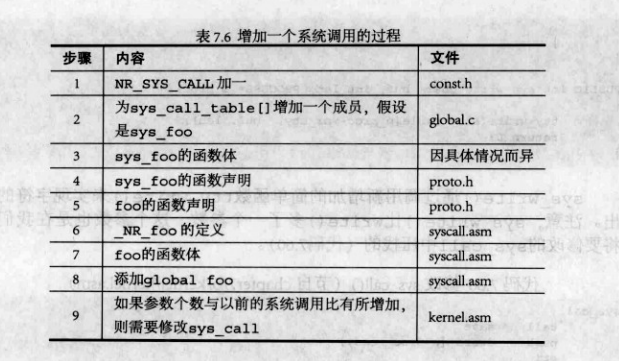
对于终端及控制台的指定问题，鉴于我已经在实验三中只保留了一个控制台，故各用户进程均指定tty0；

C:\Users\Kevin\AppData\Roaming\Tencent\Users\1269897230\QQ\WinTemp\RichOle\3UC20WC}A[VE@(HHO]TJ@PI.png



3、添加sys\_process\_sleep, sys\_disp\_str, sys\_sem\_p 以及 sys\_sem\_v系统调用：

上述过程已经完成了环境的搭建，接下来就说明实验四核心功能的实现：



（1）将const.h中的系统调用数改为5

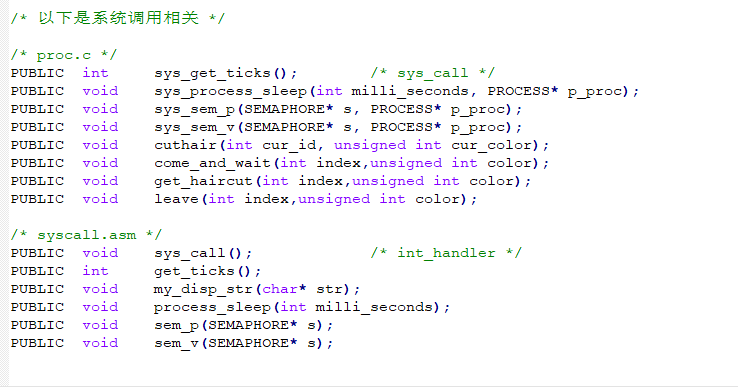
/\* system call \*/

#define NR\_SYS\_CALL 5

（2）在global.c中增加系统调用成员

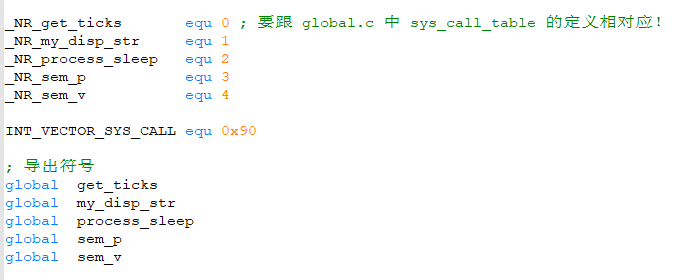
PUBLIC system\_call sys\_call\_table[NR\_SYS\_CALL] = {sys\_get\_ticks,sys\_disp\_str,sys\_process\_sleep,sys\_sem\_p,sys\_sem\_v};

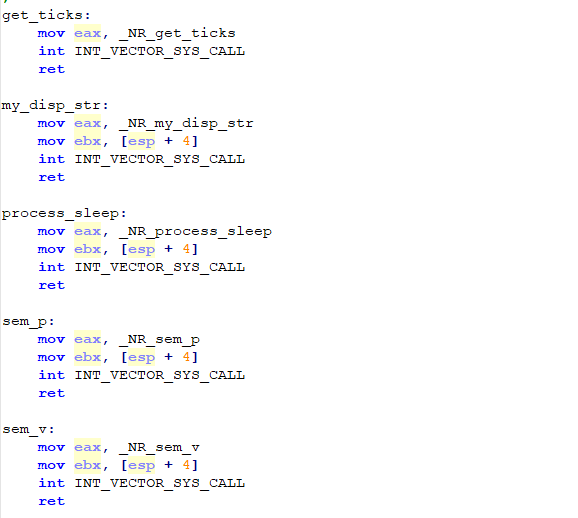
（3）在proto.h中添加函数声明



emmm......某些乱入的函数直接无视吧。。。

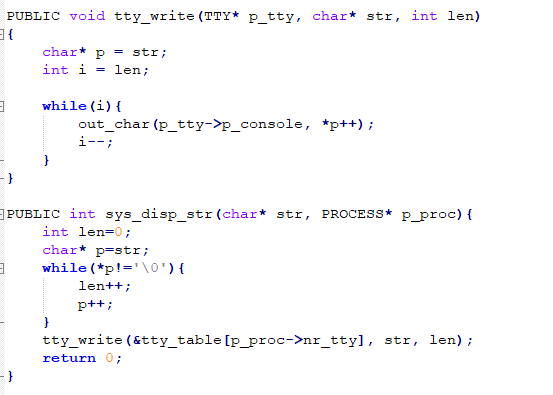
（4）在kernel/syscall.asm中增加要求系统调用



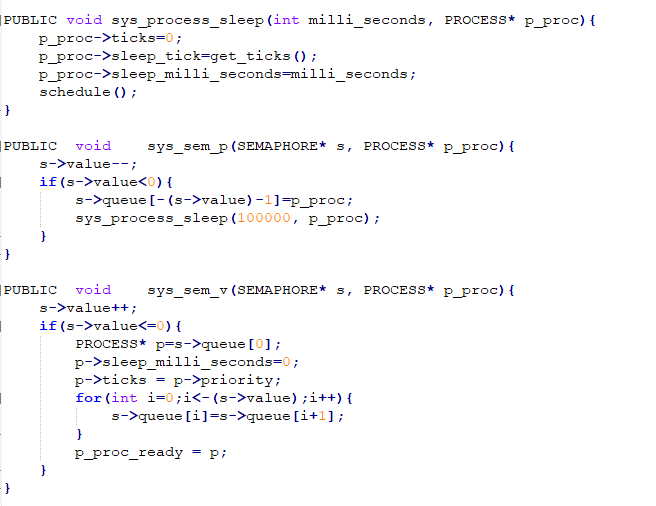


（5）增加各个系统调用的内核部分函数体

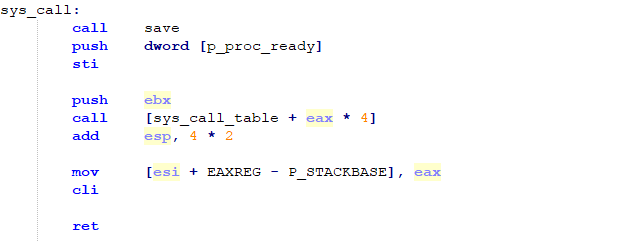
kernel/tty.c中的sys\_disp\_str:



剩余系统调用均写于kernel/proc.c中：

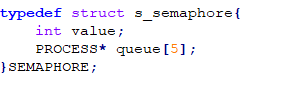


（6）由于系统调用参数发生了改变，于是对kernel/kernel.asm中的sys\_call进行修改



4、既然完成了系统调用的添加，最后就是对睡眠理发师问题的模拟了

emmm……在说明系统调用时忘了说明信号量结构体，现在此补充说明：



该结构体位于proc.h中（由于此次实验只有5个进程，故不妨将等待队列容量设为5）

接下来就是为完成模拟补充的一些属性了：

对控制台有以下两个改动：

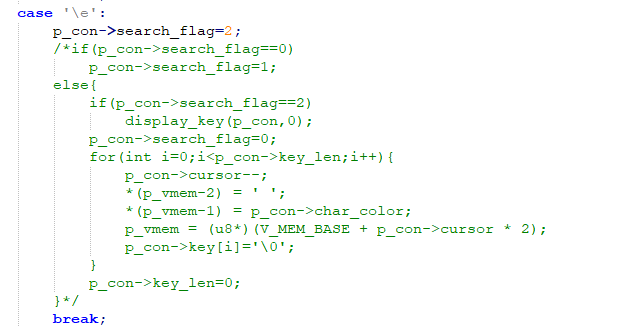
（1）console.h中给CONSOLE结构体添加属性：unsigned int char\_color;

并在console.c中的初始化操作中添加： p\_tty->p\_console->char\_color=DEFAULT\_CHAR\_COLOR;

还有将out\_char中的DEFAULT\_CHAR\_COLOR均改为p\_con->char\_color;

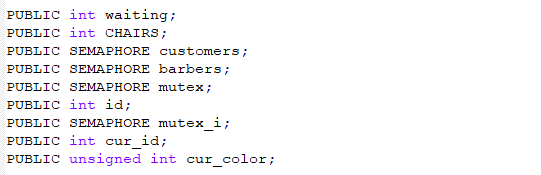
综上，即可通过char\_color属性控制console的输出颜色

（2）为了便于观察输出结果，将esc键的功能改为禁止向console打印字符

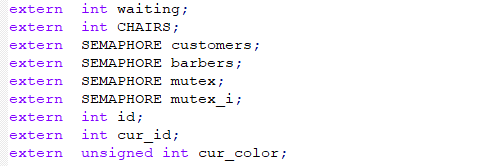


接下来就该谈谈对模拟所需变量的添加了：

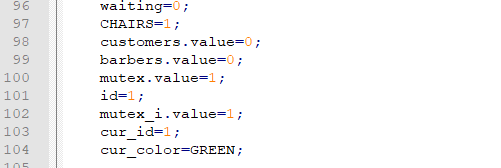
global.c:



global.h:



main.c:



waiting(等待顾客数)、CHAIRS(等待椅子数)、mutex（waiting的互斥信号量）

customers(等待顾客信号量)、barbers(就绪理发师信号量)、id（顾客id）、

mutex\_i（id的互斥信号量）、cur\_id（当前理发顾客id）、cur\_color（对应输出颜色）

最后不妨说明一些补充的函数（均写于proc.c中）

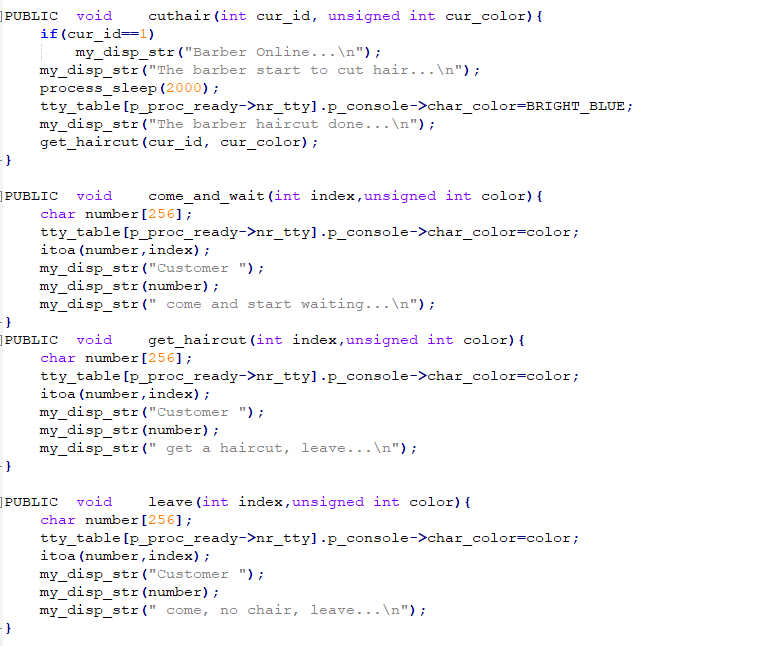
cuthair（理发师剪头发）

come\_and\_wait（顾客到来并等待）

get\_haircut（顾客剪完头发）

leave(无空位，顾客离开)

ps：cur\_id及cur\_color由得以剪头发的顾客修改



好像没别的代码添加或修改了欸，那就这样吧。。。

哦，对了，运行截图SnapShot1、2、3分别是CHAIRS为1、2、3时的程序运行情况..