1. 添加五个系统调用，每个调用输出此调用名称，不实现具体功能。

步骤1,2, P295

步骤3：

PUBLIC void sys\_process\_sleep(int milli\_sec) clock.c

PUBLIC void sys\_disp\_str(char \*str) klib.c

PUBLIC void sys\_disp\_color\_str(char \* info, int color) klib.c

PUBLIC void sys\_sem\_p(char \*name) klib.c

PUBLIC void sys\_sem\_v(char \*name) klib.c

步骤4

步骤5：proto.h

PUBLIC void user\_process\_sleep(int milli\_sec);

PUBLIC void user\_disp\_str(char \*str);

PUBLIC void user\_disp\_color\_str(char \* info, int color);

PUBLIC void user\_sem\_p(char \*name);

PUBLIC void user\_sem\_v(char \*name);

步骤6：syscall.asm

\_NR\_process\_sleep equ 2

\_NR\_disp\_str equ 3

\_NR\_disp\_color\_str equ 4

步骤7,8

1. 实现系统调用

PUBLIC void sys\_process\_sleep(int milli\_sec) clock.c

在proc.h的进程表中添加

int suspend\_time; //需要被挂起的毫秒数

int current\_ticks; //被设置挂起时的tick数

在proc.c的schedule中添加逻辑判断

在clock.c中的sys\_process\_sleep添加逻辑。

Proc.h中添加schedule项。

Schedule中判断进程是否被suspend，如果是则判断是否已经超过了规定被挂起的时间，是则挂起毫秒数归零，否则跳过此进程

1. 实现系统调用pv操作

Pv原型在klib.c中

在clock.c中加入sleep和wakeup函数，proto.h中加入原型

其中sleep的实现是在进程表中加入isSleep变量，在schedule的时候判断此进程是否isSleep，如果是则直接跳过。

Sleep函数是直接将isSleep赋值为true，并立刻调用schedule函数

Wakeup同上

Pv操作中的队列用数组简易实现

1. 再添加2个客户进程

步骤1,2,3,4,5，见书

1. 在main函数的各进程中添加Pv操作代码
2. 测试检验~完成(get后乱码的表示getcut)

顾客则显示wait

理发师剪头发cutHair

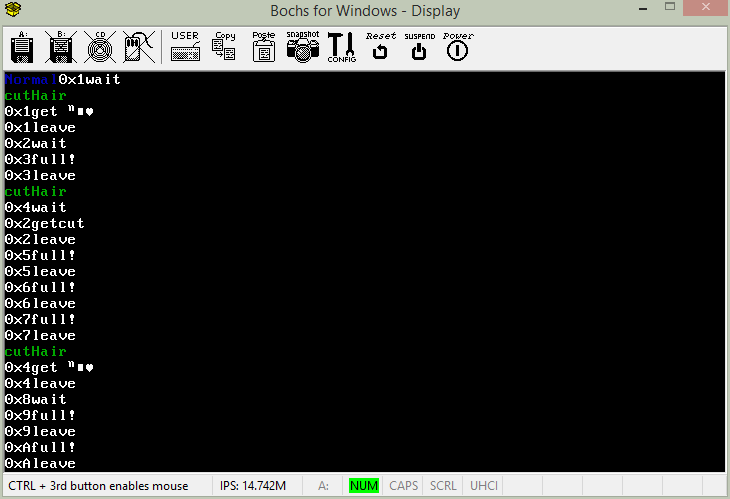
顾客看到店里满了full!

顾客离开leave

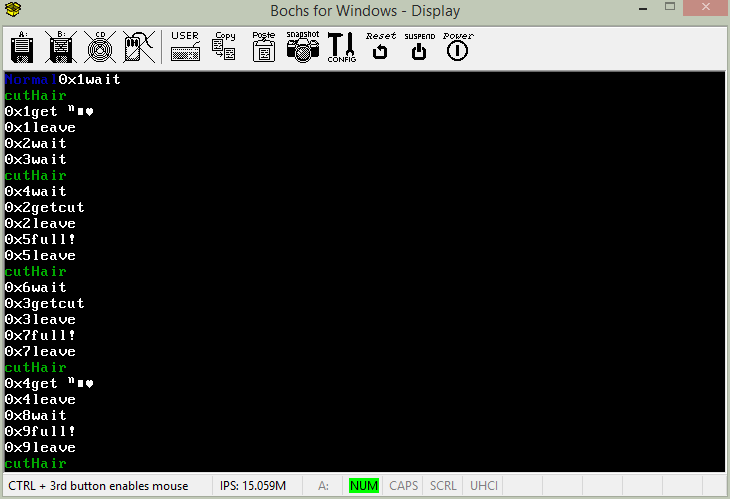
顾客得到服务getcut

注：顾客离开可能是得到了服务也可能是看到店里满了！

椅子数为1：



椅子数为2：



椅子数为3：

