JOS Lab1 做题笔记

杨健邦 515030910223

打印八进制数

Exercise 8. We have omitted a small fragment of code - the code necessary to print octal numbers using patterns of the form "%o". Find and fill in this code fragment. Remember the octal number should begin with '0'.

1. 首先了解 printf.c、console.h/c 和 printfmt.c 三个文件的关系，知道如果想在终端打印出字符，对用户提供的接口是 printf 中的 cprintf，然后将用户提供的参数（字符串格式以及对应参数）传递给库函数 printfmt.c 进行处理，然后调用 console.h 提供的接口，一个字符一个字符地输出。

2. 由于八进制数前面要多个 0，因此在 switch 中 case 等于 o 的时候先 putch（终端打印）一个 0.

3. 由于八进制数是无符号类型的，因此在获取用户提供的相应的参数时，调用的是 getuint。

4. 最后将 base 设置成 8，跳转到 number label 下，让 printnum 函数逐位打印出相应的八进制数即可。

打印符号

Exercise 9. You need also to add support for the "+" flag, which forces to precede the result with a plus or minus sign (+ or -) even for positive numbers.

1. 格式字符串的格式为`%<A><m><.p><L>X`，其中 A 为标志，也就是+所属的类型，因此在 swtich 的一开始加上一个 case +。

2. +号位只对有符号类型的数才有效。用一个 char 变量名字叫做 sign，一开始先置为 0，当读到+号时，将 sign 设置成'+'，在 case d 中检测 num 是否为负数，如果为负数则置为 '-'。对于其它无符号类型的数据，在它们的 case 中将 sign 置为 0。最后再将 sign 传进 printnum 中，如果 sign 不为 0，则需要将 sign 打印出来。（注意：直接在 case 中打印出符号会有 bug，文档最后会介绍，下面的代码是一系列测试的结果）

```
1. cprintf("正数无空格:%d\n", 1);
2. cprintf("正数有空格(一个):% d\n", 1);
3. cprintf("正数有空格(多个):%    d\n", 1);
4. cprintf("负数无空格:%d\n", -1);
5. cprintf("负数有空格:%  d\n", -1);
```

```
6. cprintf("正十六进制数有空格:%+x\n", 16);
7. cprintf("加号 0:%+d\n", 0);
8. cprintf("左对齐加符号和 0 填充:%0+18d\n", 1);
9. cprintf("左对齐加符号和 0 填充:%+18d\n", 1);

结果:
1. 正数无空格:1
2. 正数有空格(一个): 1(此处 1 左边是有一个空格的,原始代码没有考虑对空格的处理)
3. 正数有空格(多个): 1(此处 1 左边是有一个空格的,原始代码没有考虑对空格的处理)
4. 负数无空格:-1
5. 负数有空格:-1
6. 正十六进制数有空格:10(无符号数,符号不会打印出来)
7. 加号 0:+0
8. 左对齐加符号和 0 填充:+0000000000000001
9. 左对齐加符号和 0 填充:                    +1(老师给的初始代码有问题,+号会在空格填
   充左边,而不是与 1 直接相连的,这里已经做了修改)
```

打印左对齐格式的数

Exercise 10. Modify the function printnum() in lib/printfmt.c to support "%-"when printing numbers. With the directives starting with "%-", the printed number should be left adjusted. (i.e., paddings are on the right side.)

1. 因为'-'是左对齐，左对齐要先打印数字，再打印填充字符。

2. 所以重新写一个函数叫 printnnum，函数的功能就是简单地打印出相应进制的数，不打印 padding。

3. 在 printnum 中，如果 padc == '-'的话，在 printnum 中先调用 printnnum 先将所有数字打印出来（每次打印 width 都会减 1），如果 width 仍然大于 0 的话，再打印 width 个空格。(注意：左对齐的时候，只会打印空格，0 填充标志无效)

增加"n"格式符号处理

1. 利用 va_arg 可读出用户传进来的指针，然后判断是否有为空。

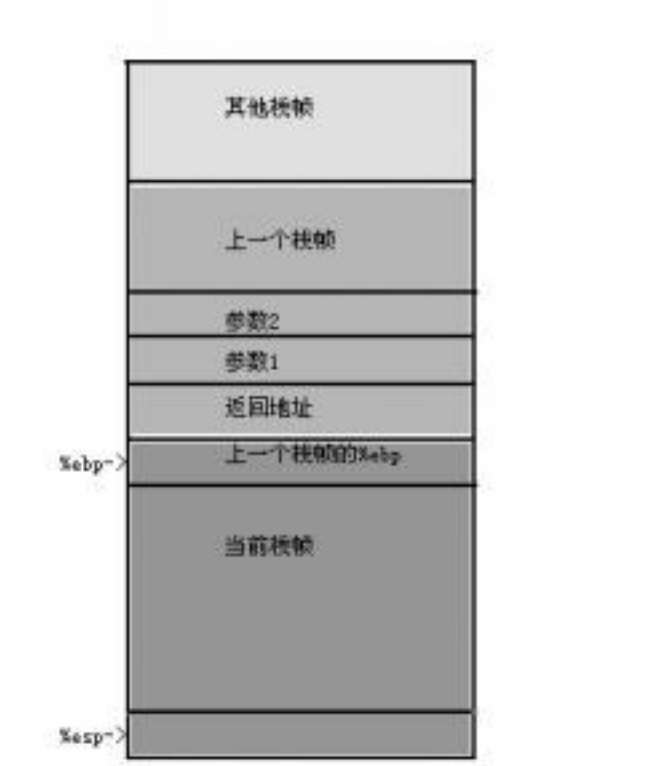
2. 从 putdat 可以知道当前已经输出了多少个字符。由于传进来的 pointer 是一个 signed char 类型的指针，而当前输出了多少字符是个正数，因此当输出的字符数大于 127 的时候则溢出。

3. 正常情况下，将 putdat 地址中的值拷贝到用户传进来的非空指针地址即可。

增加 backtrace 命令

Exercise 14. Modify your stack backtrace function to display, for each eip, the function name, source file name, and line number corresponding to that eip.

4. 先读出%ebp，由 x86 的栈结构可知，相应的%eip（返回地址）放在%ebp + 4 的地址中，而参数 1 到参数 5 分别位于%ebp + 8, %ebp + 16 … %ebp +40 的地址中。



5. 循环。而函数调用者的%ebp 正好在%ebp 这个寄存器存的地址中，因此可以循环得到 ebp，再重复上面的步骤 1。

6. 循环中止的条件。通过查看 kern/entry.S 可以知道，一开始的时候%ebp 被设置成了 0，而在其它时候，%ebp 是不可能为 0 的。因此这个是用于 backtrace 的，因此循环中止的条件是%ebp 等于 0。

7. 用二分查找的方法在符号表中找出%eip 相应的文件名，所处的代码行数，函数名，函数起始地址。

8. 将 mon_trace 函数指针和相应的名字介绍写入到 commands 数组中。

增加 time 命令

Exercise 15. In this exercise, you need to implement a rather easy "time" command. The output of the "time" is the running time (in clocks cycles) of the command. The usage of this command is like this: "time [command]".

1. 利用__asm __volatile("rdtsc":"=a"(lo),"=d"(hi))在命令执行之前和执行之后各调用一次，得到的值相减即可。

2. 但是要注意的是，连续多个 time time time time [other command]，应该只有最后一个 time 有效果，前面的 time 不产生任何作用，如果只有 time，没有其它指令，应该是不产生效果的。

发现的 BUG

在提供原始的代码中（老师提供的代码未做修改），目前发现了 vprintfmt 的三个 bug

1. 对于有符号类型的数字，其实就是 d 类型，负数是先打印负号，再调用 printnum 函数的，但是 printnum 函数中包含了打印 padding 的逻辑，因此右对齐的数字可能会出现如下情况的情况，符号和数字间有空格。

```
右对齐的负数:-          1
```

2. vprintfmt 没有处理格式化字符串中空格的情况。空格只会对有符号类型(d)的数产生影响，而且如果有了+符号，那么空格符号会被忽视。

3. 当左对齐符号'-'出现后，'0'符号会无效。