Day 19

首先我们要实现type命令，命令功能类似于cat，用于向控制台打印文件的内容。

先想想怎么获取文件的内容。还记得0x2600位置存放的文件信息的格式吗？有一个clustno可以告诉我们他从哪个扇区开始存放，而一个扇区是512字节，第一个文件处于2号扇区，位置是0x004200。据此，我们可以推算出如下公式

$$addr\_in\_image = clustno \times 512 + 0x003e00$$

再结合size字段，我们就可以把对应地址的文件给读出来了。

type指令的格式

```
type <文件名>.<扩展名>
```

我们要先在0x2600找到我们要寻找的文件，目前我们先$O(n)$的找吧。

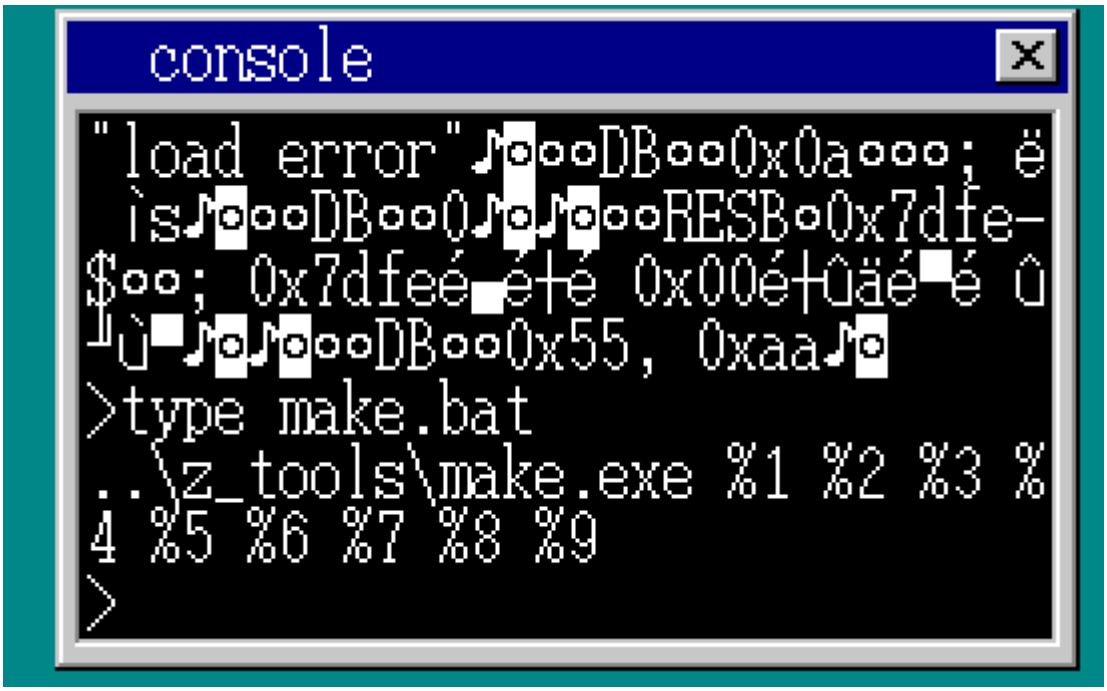
每次比较finfo[x].name与文件名的11个字节是否全部相等，相等则找到，全不相等则找不到这样的文件

```
for (y = 0; y < 11; y++) {
    s[y] = ' ';
}
y = 0;
for (x = 5; y < 11 && cmdline[x] != 0; x++) {
    if (cmdline[x] == '.' && y <= 8) {
        y = 8;
    } else {
        s[y] = cmdline[x];
        if ('a' <= s[y] && s[y] <= 'z') {
            s[y] -= 0x20;    // 转换成大写
        }
        y++;
    }
}
for (x = 0; x < 224; x++) {
    if (finfo[x].name[0] == 0x00) {
        break;
    }
    if ((finfo[x].type & 0x18) == 0) {
        for (y = 0; y < 11; y++) {
            if (finfo[x].name[y] != s[y]) {
                continue;
            }
        }
        break;
    }
}
if (x < 224 && finfo[x].name[0] != 0x00) {
```

```
    // 找到了!
    y = finfo[x].size;
    p = (char *) (finfo[x].clustno * 512 + 0x003e00 + ADR_DISKIMG);
    cursor_x = 8;
    for (x = 0; x < y; x++) {
        // 打印文件内容
        s[0] = p[x];
        s[1] = 0;
        putfonts8_asc_sht(sheet, cursor_x, cursor_y, COL8_FFFFFF, COL8_000000, s, 1);
        cursor_x += 8;
        if (cursor_x == 8 + 240) { // 打满一行自动换行
            cursor_x = 8;
            cursor_y = cons_newline(cursor_y, sheet);
        }
    }
} else {
    // 没找到
    putfonts8_asc_sht(sheet, 8, cursor_y, COL8_FFFFFF, COL8_000000, "File not found.", 15);
    cursor_y = cons_newline(cursor_y, sheet);
}
```

命令判定部分

```
else if (memcmp(cmdline, "type", 4) == 0) { // 作者这里用的是丑丑的多重条件
    // ..........
}
```



上面的乱码是`type ipl10.nas`的结果

我们接下来要处理特殊字符

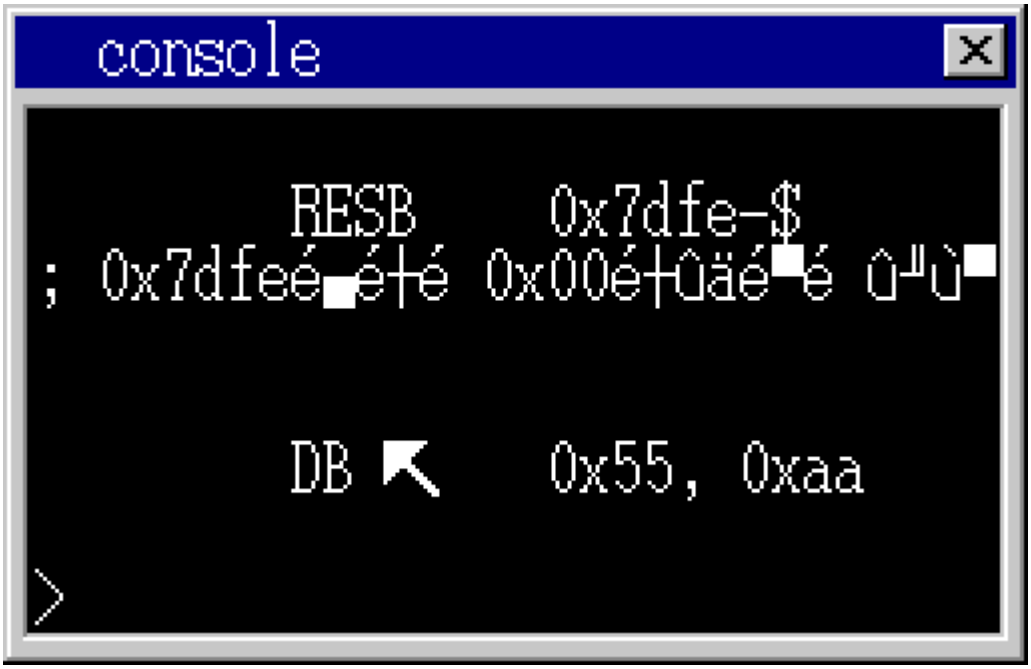
ASCII	名字	处理方法
0x09	制表符	显示空格直到x被4整除
0x0a	换行符	换行
0x0d	回车符	不管他

```
if (s[0] == 0x09) { // 制表符
    for (;;) {
        putfonts8_asc_sht(sheet, cursor_x, cursor_y, COL8_FFFFFF,
                COL8_000000, " ", 1);
        cursor_x += 8;
        if (cursor_x == 8 + 240) {
            cursor_x = 8;
            cursor_y = cons_newline(cursor_y, sheet);
        }
        if (((cursor_x - 8) & 0x1f) == 0) { // (cursor_x-8) % 32
            break;
        }
    }
} else if (s[0] == 0x0a) { // 换行符
    cursor_x = 8;
    cursor_y = cons_newline(cursor_y, sheet);
} else if (s[0] == 0x0d) {
    // do nothing
} else { // 普通字符
    putfonts8_asc_sht(sheet, cursor_x, cursor_y, COL8_FFFFFF,
            COL8_000000, s, 1);
    cursor_x += 8;
    if (cursor_x == 8 + 240) {
        cursor_x = 8;
        cursor_y = cons_newline(cursor_y, sheet);
    }
     }
```

以上是特殊字符判断部分

为什么要将cursor_x减8呢？因为命令行窗口的边框有8个像素，所以要把那部分给去 掉。然后，1个字符的宽度是8个像素，每个制表位相隔4个字符。4x宽度（8）=32。所以要对32取模。

再次测试



除了日文还乱码之外，其他已经正常了许多了。

接下来引入对fat的支持。

众所周知，很多文件都不止512字节，一个扇区是不足以保存他们的，那么我们该如何处理呢？我们需要读取fat(file allocation table)

fat存储在0x00200~0x013ff，我们先给他读进内存。在正式使用它之前，需要先进行解压缩

以三个字节为一组，进行4个bit为单位的换序

F0 FF FF → FF0 FFF

ab cd ef → dab efc

```
void file_readfat(int *fat, unsigned char *img)
{
    int i, j = 0;
    for (i = 0; i < 2880; i += 2) {
        fat[i + 0] = (img[j + 0]      | img[j + 1] << 8) & 0xfff;
        fat[i + 1] = (img[j + 1] >> 4 | img[j + 2] << 4) & 0xfff;
        j += 3;
    }
    return;
}
```

该函数将img地址未经解压的fat解压缩并存入fat指针指向的位置

然后我们就可以查这个fat表了，他其实相当于一个链表，每当你访问一个扇区，你就访问对应的fat，从中你可以得知你应该访问的下一个扇区，以此类推，直到fat中指示的下一个扇区为0xff8~0xfff的值，则说明文件结束。

接下来我们写一个函数用于读入文件到内存中，只要当读完一个扇区且文件还没读完，按照fat表进行跳转就ok了

```
void file_loadfile(int clustno, int size, char *buf, int *fat, char *img)
{
    int i;
    for (;;) {
        if (size <= 512) {
            for (i = 0; i < size; i++) {
                buf[i] = img[clustno * 512 + i];
            }
            break;
        }
        for (i = 0; i < 512; i++) {
            buf[i] = img[clustno * 512 + i];
        }
        size -= 512;
        buf += 512;
        clustno = fat[clustno];
    }
    return;
}
```

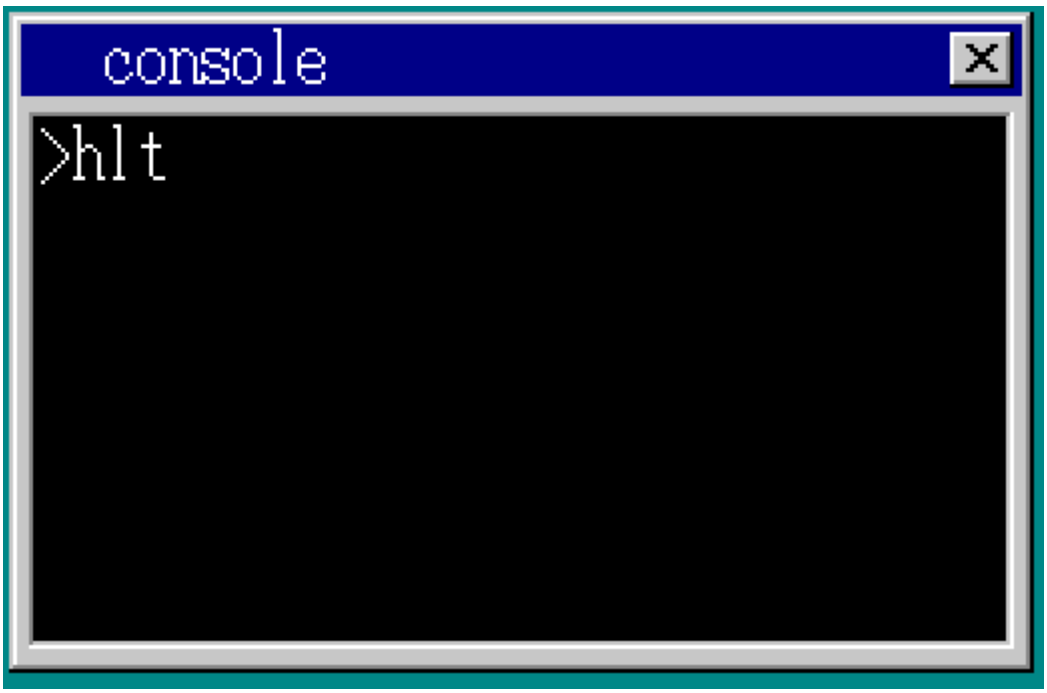
能够读取文件了，接下来我们想办法引入对应用程序的支持。

主要思路是先编写程序，然后用nask编译成文件，然后添加到镜像当中去。将文件load到另一个连续的内存区域，然后farjump到那个内存区域就ok了

下段程序运行hlt.hrb

```
strncpy(s, "HLT     HRB", 11);
for (x = 0; x < 224 && finfo[x].name[0]; x++) {
    if ((finfo[x].type & 0x18) == 0) {
        for (y = 0; y < 11; y++) {
            if (finfo[x].name[y] != s[y]) {
                continue;
            }
        }
        break;
    }
}
if (x < 224 && finfo[x].name[0]) {
    p = (char *) memman_alloc_4k(memman, finfo[x].size);
    file_loadfile(finfo[x].clustno, finfo[x].size, p, fat, (char *)
            (ADR_DISKIMG + 0x003e00));
    set_segmdesc(gdt + 1003, finfo[x].size - 1, (int) p, AR_CODE32_ER);
    farjmp(0, 1003 * 8);
    memman_free_4k(memman, (int) p, finfo[x].size);
```

```
} else {
    putfonts8_asc_sht(sheet, 8, cursor_y, COL8_FFFFFF, COL8_000000, "File not found.", 15);
    cursor_y = cons_newline(cursor_y, sheet);
}
cursor_y = cons_newline(cursor_y, sheet);
```



成功，今天到此为止