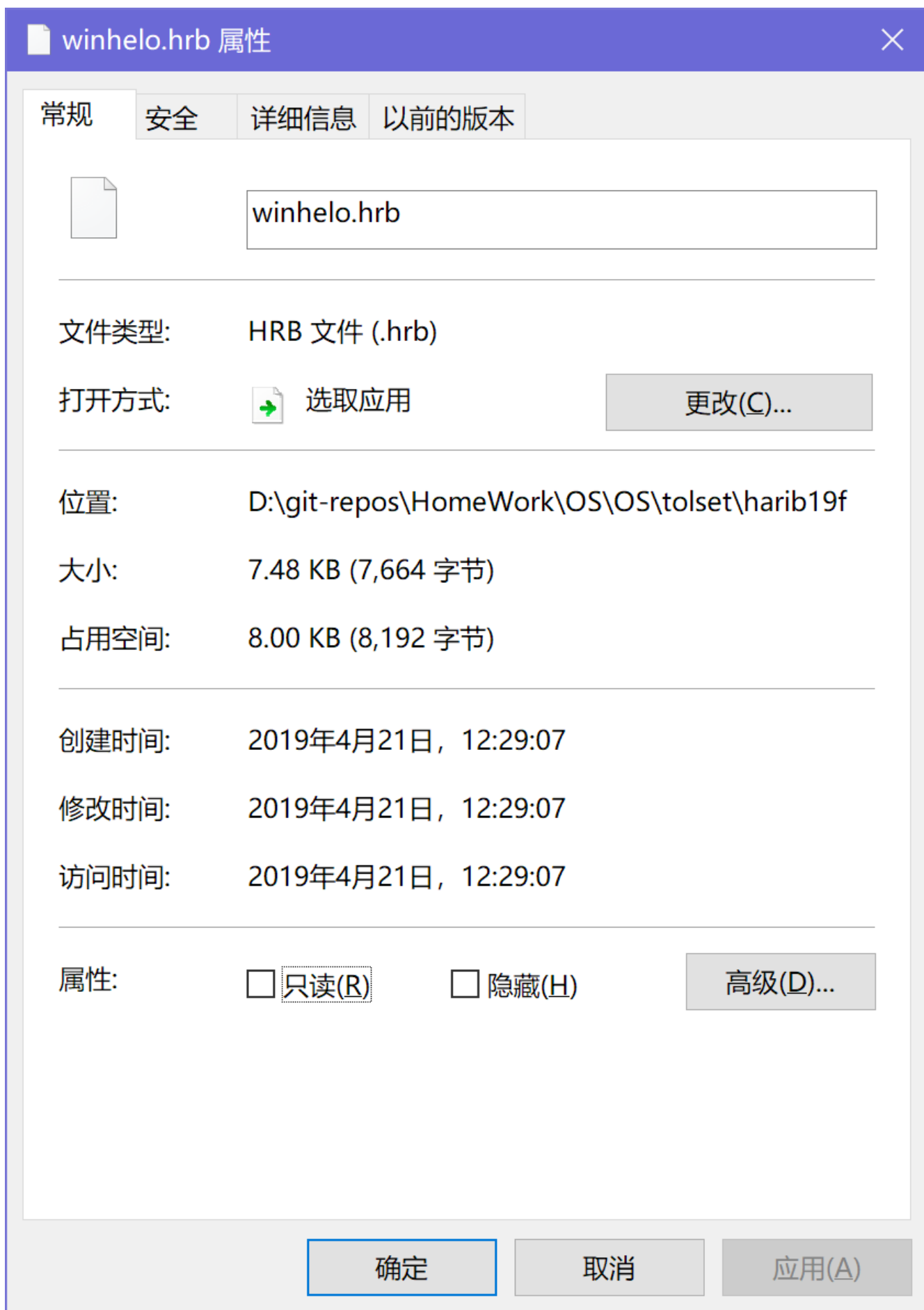


Day 23



这个是我们之前实验当中winhelo程序的详细信息，可以看到这个文件达到了惊人的7.48KB大小。使用二进制查看器

00000000	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000000	00	30	00	00	48	61	72	69	00	00	00	00	00	04	00	00	.0...Har i.....
00000010	5c	1d	00	00	94	00	00	00	00	00	00	e9	6b	00	00	00	\...?...聞...
00000020	60	21	00	00	55	89	e5	68	00	04	00	00	6a	ff	6a	32	`!...U 文 h...j j 2
00000030	68	96	00	00	00	68	10	04	00	00	e8	2a	00	00	00	83	h?...h...?...麵
00000040	c4	14	c9	e9	1a	00	00	00	ba	01	00	00	00	8a	44	24	. 硃? ... 套 \$
00000050	04	cd	40	c3	53	ba	02	00	00	00	8b	5c	24	08	cd	40	. 蚬 胚 ? ... 嫵 \$. 蚬
00000060	5b	c3	ba	04	00	00	00	cd	40	57	56	53	ba	05	00	00	[煤 蚬 WVS? ...
00000070	00	8b	5c	24	10	8b	74	24	14	8b	7c	24	18	8b	44	24	. 嫵 \$. 嫵 \$. 嫵 \$. 婦 \$
00000080	1c	8b	4c	24	20	cd	40	5b	5e	5f	c3	55	89	e5	5d	e9	. 嫵 \$ 蚬 [^ _ 肱 文] 聞
00000090	90	ff	ff	ff	68	65	6c	6c	6f	00	00	00	00	00	00	00	h e l l o
000000a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

可以看到这个文件当中有大量的00，这显然有很多是没有用的。这些0是哪里来的呢？

原因在于winhelo2.c的 `char buf[150 * 50];` 这相当于在可执行文件中插入了 $150 \times 50 = 7500$ 个字节的00。

如果我们的操作系统可以在运行的时候动态给程序分配内存的话就好了。我们仿照c语言的命名，把这个api叫做 `malloc`。

`api_malloc`

`malloc`不可以直接调用操作系统的`memman_malloc`，这是因为`mamman_malloc`分配的空间是在系统段当中，应用程序是无法访问这个区域的。

作者给出了一个什么样的解决方案呢？应用程序提前指定好自己会使用的最大内存量，提前给他准备好，当应用程序请求分配内存的时候就从这段内存当中拿出来一部分给他就好了。

在哪里给定这样的值呢？作者写的**bim2hrb**工具接受的第三个参数可以进行指定

```
$(BIM2HRB) <filename>.bim <filename>.hrb <预存空间大小>
```

而这个指定的参数，则保存在hrb文件的0x0020处

`mamman`设计

初始化api

EDX	8
EBX	memman的地址
EAX	memman所管理的内存空间的起始地址
ECX	memman所管理的内存空间的字节数

分配api

EDX	9
EBX	memman的地址
ECX	需要请求的字节数
EAX	分配到的内存空间地址

释放api

EDX	10
EBX	memman的地址
EAX	需要释放的内存空间地址
ECX	需要释放的字节数

hrb_api修改的部分

```

else if (edx == 8) {
    memman_init((struct MEMMAN *) (ebx + ds_base));
    ecx &= 0xffffffff0; /*以16字节为单位*/
    memman_free((struct MEMMAN *) (ebx + ds_base), eax, ecx);
} else if (edx == 9) {
    ecx = (ecx + 0x0f) & 0xffffffff0; /*以16字节为单位进位取整*/
    reg[7] = memman_alloc((struct MEMMAN *) (ebx + ds_base), ecx);
} else if (edx == 10) {
    ecx = (ecx + 0x0f) & 0xffffffff0; /*以16字节为单位进位取整*/
    memman_free((struct MEMMAN *) (ebx + ds_base), eax, ecx);
}

```



再整个画点的

EDX	11
EBX	窗口句柄
ESI	显示位置的x坐标
EDI	显示位置的y坐标
EAX	色号

```

} else if (edx == 11) {
    sht = (struct SHEET *) ebx;
    sht->buf[sht->bysize * edi + esi] = eax;
    sheet_refresh(sht, esi, edi, esi + 1, edi + 1);
}

```

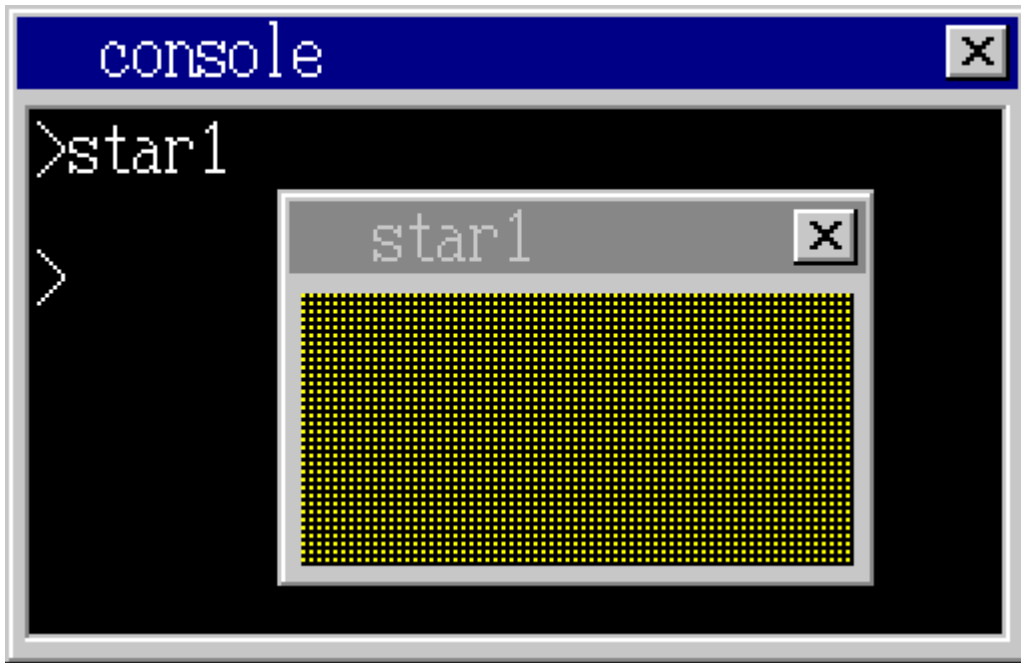
测试程序

```

void HariMain(void)
{
    char *buf;
    int win, y, x;
    api_initmalloc();
    buf = api_malloc(150 * 100);
    win = api_openwin(buf, 150, 100, -1, "star1");
    api_boxfillwin(win, 6, 26, 143, 93, 0 /* 黒 */);
    for (y = 26; y <= 93; y += 2) {
        for (x = 6; x <= 143; x += 2) {
            api_point(win, x, y, 3);
        }
    }
    api_end();
}

```

```
}
```



我们发现，每次画点都会进行一次窗体刷新。这种做法效率是很低的，因为我们每次更新一个像素点，却要重新绘制整个窗口区域。可以考虑接受一个参数，从而决定是否在画点之后进行窗体刷新，并且设计一个新的刷新窗口的API

EDX	12
EBX	窗口句柄
EAX	x0
ECX	y0
ESI	x1
EDI	y1

```
else if (edx == 11) {
    if ((ebx & 1) == 0) {
        sht->buf[((struct SHEET *) (ebx)) -> bsize * edi + esi] = eax;
        sheet_refresh((struct SHEET *) (ebx), esi, edi, esi + 1, edi + 1);
    } else {
        sht->buf[((struct SHEET *) (ebx ^ 1)) -> bsize * edi + esi] = eax;
        sheet_refresh((struct SHEET *) (ebx ^ 1), esi, edi, esi + 1, edi + 1);
    }
} else if (edx == 12) {
    sht = (struct SHEET *) ebx;
    sheet_refresh(sht, eax, ecx, esi, edi);
}
```

为了节省参数，可以把这个信息融入到窗口句柄中。struct SHEET的地址一定是一个偶数，那么我们可以让程序在指定一个奇数（即在原来的数值上加1）的情况下不进行自动刷新。

```
void HariMain(void)
{
    char *buf;
    int win, y, x;
    api_initmalloc();
    buf = api_malloc(150 * 100);
    win = api_openwin(buf, 150, 100, -1, "star1");
    api_boxfilwin(win, 6, 26, 143, 93, 0 /* 黒 */);
    for (y = 26; y <= 93; y += 2) {
        for (x = 6; x <= 143; x += 2) {
            api_point(win, x, y, 3);
        }
    }
    api_refreshwin(win, 6, 26, 144, 94);
    api_end();
}
```

运行正常

画直线

EDX	13
EBX	窗口句柄
EAX	x0
ECX	y0
ESI	x1
EDI	y1
EBP	色号

```
} else if (edx == 13) {
    sht = (struct SHEET *) (ebx & 0xfffffffffe);
    hrb_api_linewin(sht, eax, ecx, esi, edi, ebp);
    if ((ebx & 1) == 0) {
        sheet_refresh(sht, eax, ecx, esi + 1, edi + 1);
    }
}
```

```
void hrb_api_linewin(struct SHEET *sht, int x0, int y0, int x1, int y1, int col)
{
    int i, x, y, len, dx, dy;
    dx = x1 - x0;
    dy = y1 - y0;
```

```

x = x0 << 10;
y = y0 << 10;
if (dx < 0) {
    dx = - dx;
}
if (dy < 0) {
    dy = - dy;
}
if (dx >= dy) {
    len = dx + 1;
    if (x0 > x1) {
        dx = -1024;
    } else {
        dx = 1024;
    }
    if (y0 <= y1) {
        dy = ((y1 - y0 + 1) << 10) / len;
    } else {
        dy = ((y1 - y0 - 1) << 10) / len;
    }
} else {
    len = dy + 1;
    if (y0 > y1) {
        dy = -1024;
    } else {
        dy = 1024;
    }
    if (x0 <= x1) {
        dx = ((x1 - x0 + 1) << 10) / len;
    } else {
        dx = ((x1 - x0 - 1) << 10) / len;
    }
}
for (i = 0; i < len; i++) {
    sht->buf[(y >> 10) * sht->bysize + (x >> 10)] = col;
    x += dx;
    y += dy;
}
return;
}

```

```

int api_openwin(char *buf, int xsiz, int ysiz, int col_inv, char *title);
void api_initmalloc(void);
char *api_malloc(int size);
void api_refreshwin(int win, int x0, int y0, int x1, int y1);
void api_linewin(int win, int x0, int y0, int x1, int y1, int col);
void api_end(void);
void HariMain(void)
{
    char *buf;
    int win, i;
    api_initmalloc();
    buf = api_malloc(160 * 100);
}

```

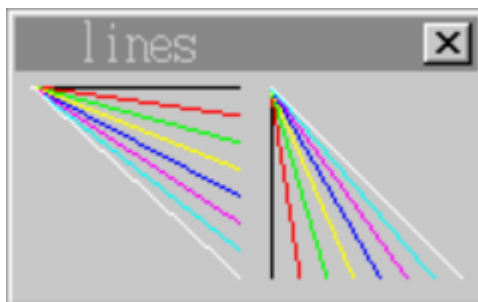


```

win = api_openwin(buf, 160, 100, -1, "lines");
for (i = 0; i < 8; i++) {
    api_linewin(win + 1, 8, 26, 77, i * 9 + 26, i);
    api_linewin(win + 1, 88, 26, i * 9 + 88, 89, i);
}
api_refreshwin(win, 6, 26, 154, 90);
api_end();
}

```

略去了汇编语言的部分QAQ



关闭窗口

应用程序结束以后，窗口也应当清除

EDX	14
EBX	窗口句柄

```

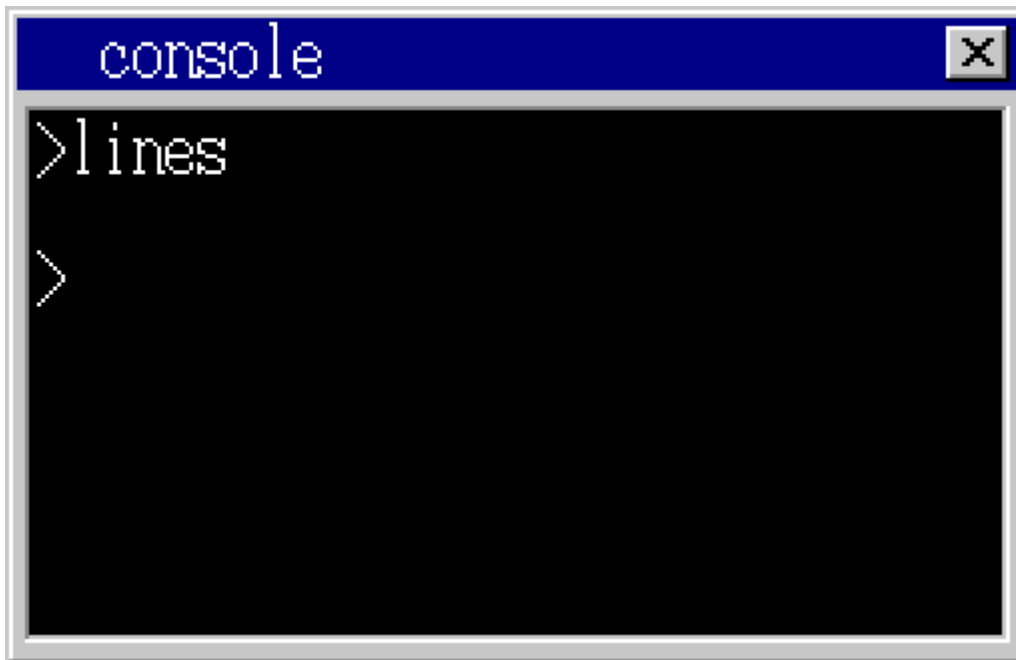
} else if (edx == 14) {
    sheet_free((struct SHEET *) ebx);
}

```

```

void HariMain(void)
{
    char *buf;
    int win, i;
    api_initmalloc();
    buf = api_malloc(160 * 100);
    win = api_openwin(buf, 160, 100, -1, "lines");
    for (i = 0; i < 8; i++) {
        api_linewin(win + 1, 8, 26, 77, i * 9 + 26, i);
        api_linewin(win + 1, 88, 26, i * 9 + 88, 89, i);
    }
    api_refreshwin(win, 6, 26, 154, 90);
    api_closewin(win); /*这里! */
    api_end();
}

```



键盘输入

EDX	15
EAX	0.....没有键盘输入时返回□1，不休眠
	1.....休眠直到发生键盘输入
EAX	输入的字符编码

```
else if (edx == 15) {
    for (;;) {
        io_cli();
        if (fifo32_status(&task->fifo) == 0) {
            if (eax != 0) {
                task_sleep(task); /* FIFO为空，休眠并等待*/
            } else {
                io_sti();
                reg[7] = -1;
                return 0;
            }
        }
        i = fifo32_get(&task->fifo);
        io_sti();
        if (i <= 1) { /*光标用定时器*/
            /*应用程序运行时不需要显示光标，因此总是将下次显示用的值置为1*/
            timer_init(cons->timer, &task->fifo, 1); /*下次置为1*/
            timer_settime(cons->timer, 50);
        }
        if (i == 2) { /*光标ON */
            cons->cur_c = COL8_FFFFFFFF;
        }
    }
}
```

```

    }
    if (i == 3) { /*光标OFF */
        cons->cur_c = -1;
    }
    if (256 <= i && i <= 511) { /*键盘数据 (通过任务A) */
        reg[7] = i - 256;
        return 0;
    }
}
}

```

我们将定时器放入了console当中

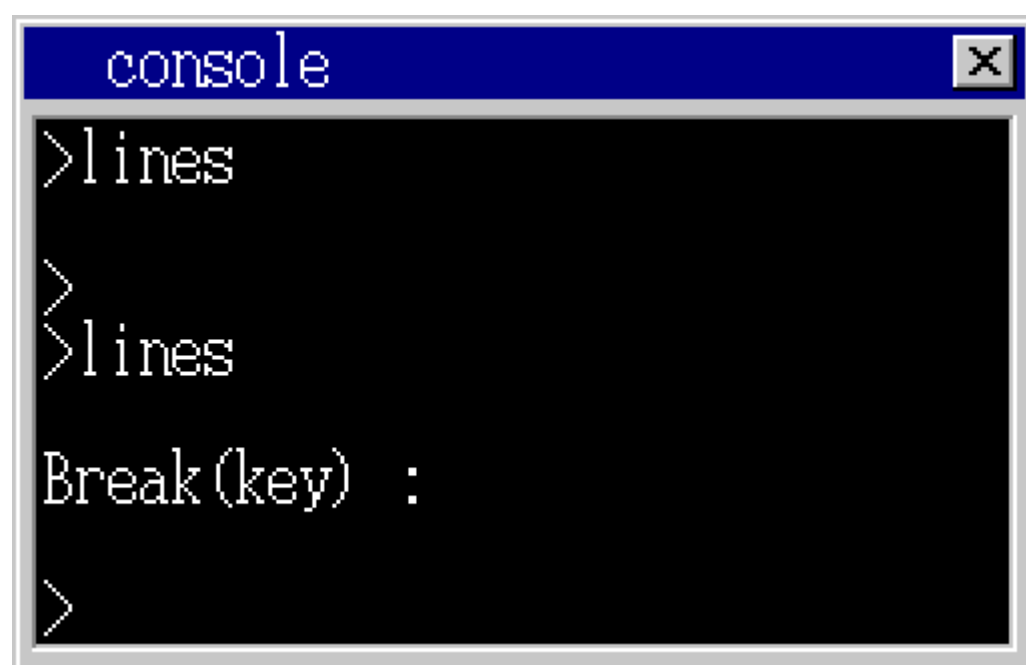
```

void api_closewin(int win);
int api_getkey(int mode);
void api_end(void);
void HariMain(void)
{
    (中略)
    api_refreshwin(win, 6, 26, 154, 90);
    for (;;) {
        if (api_getkey(1) == 0x0a) {
            break;
        }
    }
    api_closewin(win);
    api_end();
}

```

我们等待回车键，回车键按下后再结束程序。

对了。如果用的是 Shift + F1 结束程序的话，窗口还是不会自动消失，这不够好。我们可以在sheet中添加一个task字段，里面写了它对应的应用程序。我们结束程序的时候检测以下所有sheet，如果有属于这个应用程序的sheet，那么我们就把它free掉。



```
>lines
```

```
>
```

```
>lines
```

```
Break(key) :
```

```
>
```