Day 29

今天我们为操作系统引入原生的压缩文件支持，使用作者自己鼓捣出来的tek格式。

edimg中使用的压缩格式就是tek。edimg的源码在autodec_.c中，我们直接C-cv相应的代码

我们整理一下解压缩函数

```
int tek_getsize(unsigned char *p)
{
    static char header[15] = {
        0xff, 0xff, 0xff, 0x01, 0x00, 0x00, 0x00, 0x4f, 0x53, 0x41, 0x53, 0x4b, 0x43, 0x4d,
0x50
    };
    int size = -1;
    if (memcmp(p + 1, header, 15) == 0 && (*p == 0x83 || *p == 0x85 || *p == 0x89)) {
        p += 16;
        size = tek_getnum_s7s(&p);
    }
    return size;
}
int tek_decomp(unsigned char *p, char *q, int size)
{
    int err = -1;
    if (*p == 0x83) {
        err = tek_decode1(size, p, q);
    } else if (*p == 0x85) {
        err = tek_decode2(size, p, q);
    } else if (*p == 0x89) {
        err = tek_decode5(size, p, q);
    }
    if (err != 0) {
        return -1;
    }
    return 0;
}
```

我们把file_loadfile包装一下，以实现自动解压。

```
char *file_loadfile2(int clustno, int *psize, int *fat)
{
    int size = *psize, size2;
    struct MEMMAN *memman = (struct MEMMAN *) MEMMAN_ADDR;
    char *buf, *buf2;
    buf = (char *) memman_alloc_4k(memman, size);
    file_loadfile(clustno, size, buf, fat, (char *) (ADR_DISKIMG + 0x003e00));
    if (size >= 17) {
        size2 = tek_getsize(buf);
        if (size2 > 0) {
```

```
            buf2 = (char *) memman_alloc_4k(memman, size2);
            tek_decomp(buf, buf2, size2);
            memman_free_4k(memman, (int) buf, size);
            buf = buf2;
            *psize = size2;
        }
    }
    return buf;
}
```

psize用来返回解压缩后的文件大小。

实现了自动压缩功能，我们就可以把我们的文件压缩之后再放入img中了

```
bim2bin -osacmp in:nihongo.org out:nihongo.fnt -tek2
```

其他文件同理。

为了实现原生支持，我们要把系统中所有file_loadfile的调用改成file_loadfile2

- cmd_app
- hrb_api

实现标准函数。c语言中提供了许多标准函数，例如

- printf
- putchar
- strcmp
- malloc

等

```
#include "apilib.h"
int putchar(int c)
{
    api_putchar(c);
    return c;
}
```

```
#include "apilib.h"
void exit(int status)
{
    api_end();
}
```

```

```

```c
#include <stdio.h>
#include <stdarg.h>
#include "apilib.h"
int printf(char *format, ...)
{
    va_list ap;
    char s[1000];
    int i;
    va_start(ap, format);
    i = vsprintf(s, format, ap);
    api_putstr0(s);
    va_end(ap);
    return i;
}
```

```c
void *malloc(int size)
{
    char *p = api_malloc(size + 16);
    if (p != 0) {
        *((int *) p) = size;
        p += 16;
    }
    return p;
}
```

```c
void free(void *p)
{
    char *q = p;
    int size;
    if (q != 0) {
        q -= 16;
        size = *((int *) q);
        api_free(q, size + 16);
    }
    return;
}
```

这个16是干什么的呢？为了实现只提供一个指针就可以free，我们把大小存在分配的开头的4个字节中，返回的指针是从第17个字节开始的。

于是释放的时候只要取p之前的16个字节的前4个字节，就可以得知要释放的内存到底有多大，然后就可以释放了。

使用16而不是4的原因是内存地址为16字节的倍数时，CPU的处理速度有时候可以更快

实现非矩形窗口，由于我们之前就设计了透明色机制，我们直接使用透明色就可以实现非矩形窗口了。

测试更多应用程序

```
/*bball.c*/
#include "apilib.h"
void HariMain(void)
{
    int win, i, j, dis;
    char buf[216 * 237];
    struct POINT {
        int x, y;
    };
    static struct POINT table[16] = {
        { 204, 129 }, { 195, 90 }, { 172, 58 }, { 137, 38 }, { 98, 34 },
        { 61, 46 }, { 31, 73 }, { 15, 110 }, { 15, 148 }, { 31, 185 },
        { 61, 212 }, { 98, 224 }, { 137, 220 }, { 172, 200 }, { 195, 168 },
        { 204, 129 }
    };
    win = api_openwin(buf, 216, 237, -1, "bball");
    api_boxfilwin(win, 8, 29, 207, 228, 0);
    for (i = 0; i <= 14; i++) {
        for (j = i + 1; j <= 15; j++) {
            dis = j - i; /*两点间的距离*/
            if (dis >= 8) {
                dis = 15 - dis; /*逆向计数*/
            }
            if (dis != 0) {
                api_linewin(win, table[i].x, table[i].y, table[j].x, table[j].y, 8 - dis);
            }
        }
    }
    for (;;) {
        if (api_getkey(1) == 0x0a) {
            break; /*按下回车键则break; */
        }
    }
    api_end();
}
```

运行发现一些线条显示不正常，是refresh的bug导致的，我们修改一下划线的api，swap以下大小参数

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

```
                edi = i;
            }
            sheet_refresh(sht, eax, ecx, esi + 1, edi + 1);
        }
    }
```

还有外星人游戏，这一段与操作系统相关性不大，就跳过去了