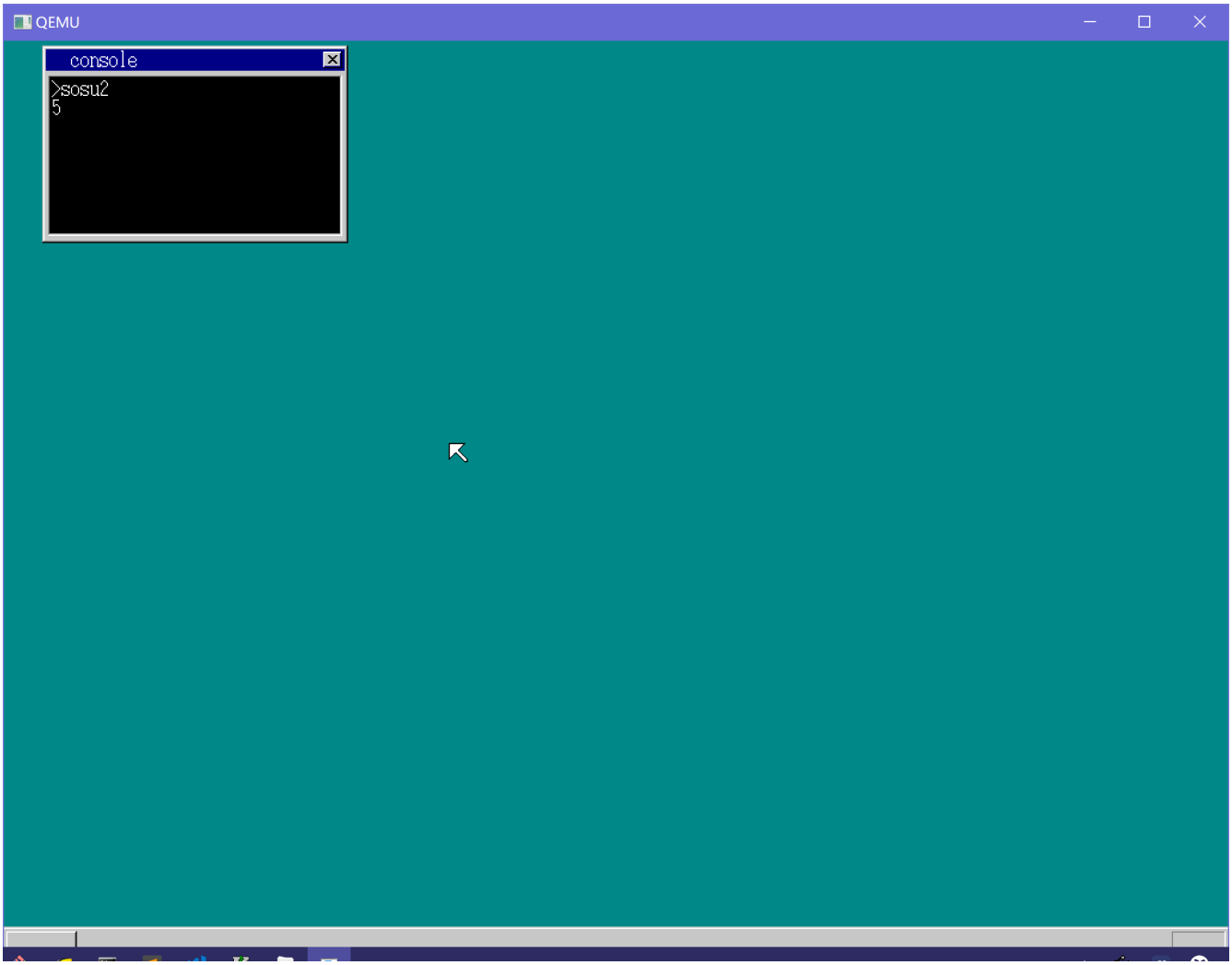
Day 28

先来一个简单的程序

```
#include <stdio.h>
#include "apilib.h"
#define MAX 1000
void HariMain(void)
{
    char flag[MAX], s[8];
    int i, j;
    for (i = 0; i < MAX; i++) {
        flag[i] = 0;
    }
    for (i = 2; i < MAX; i++) {
        if (flag[i] == 0) {
            sprintf(s, "%d ", i);
            api_putstr0(s);
            for (j = i * 2; j < MAX; j += i) {
                flag[j] = 1;
            }
        }
    }
    api_end();
}
```

这段程序是求1000以内的素数的，当我们尝试扩大范围哦，将MAX改为10000后，程序无法正常的运行了。



输出5之后程序就不动了。

注意到编译的时候有一条警告：`Warning : can't link __alloca`

看来与他有关。为什么会这样呢？这是因为c语言编译器规定，如果栈中的变量超过4KB，就需要调用__alloca函数去获取栈中的空间

编写__alloca函数以供链接和调用

```
[FORMAT "WCOFF"]
[INSTRSET "i486p"]
[BITS 32]
[FILE "alloca.nas"]

GLOBAL __alloca

[SECTION .text]

__alloca:
        ADD EAX,-4
        SUB ESP,EAX
        JMP DWORD [ESP+EAX] ; 代替RET
```

那么什么时候__alloca会被调用呢？

- 要执行的操作从栈中分配EAX个字节的内存空间（ESP -= EAX;）
- 要遵守的规则不能改变ECX、EDX、EBX、EBP、ESI、EDI的值（可以临时改变它们的 值，但要使用PUSH/POP来复原）

以下__alloca是错误的

```
SUB ESP,EAX
RET
```

RET相当于POP EIP，ESP的值被改变了，于是EIP被装入了错误的返回地址

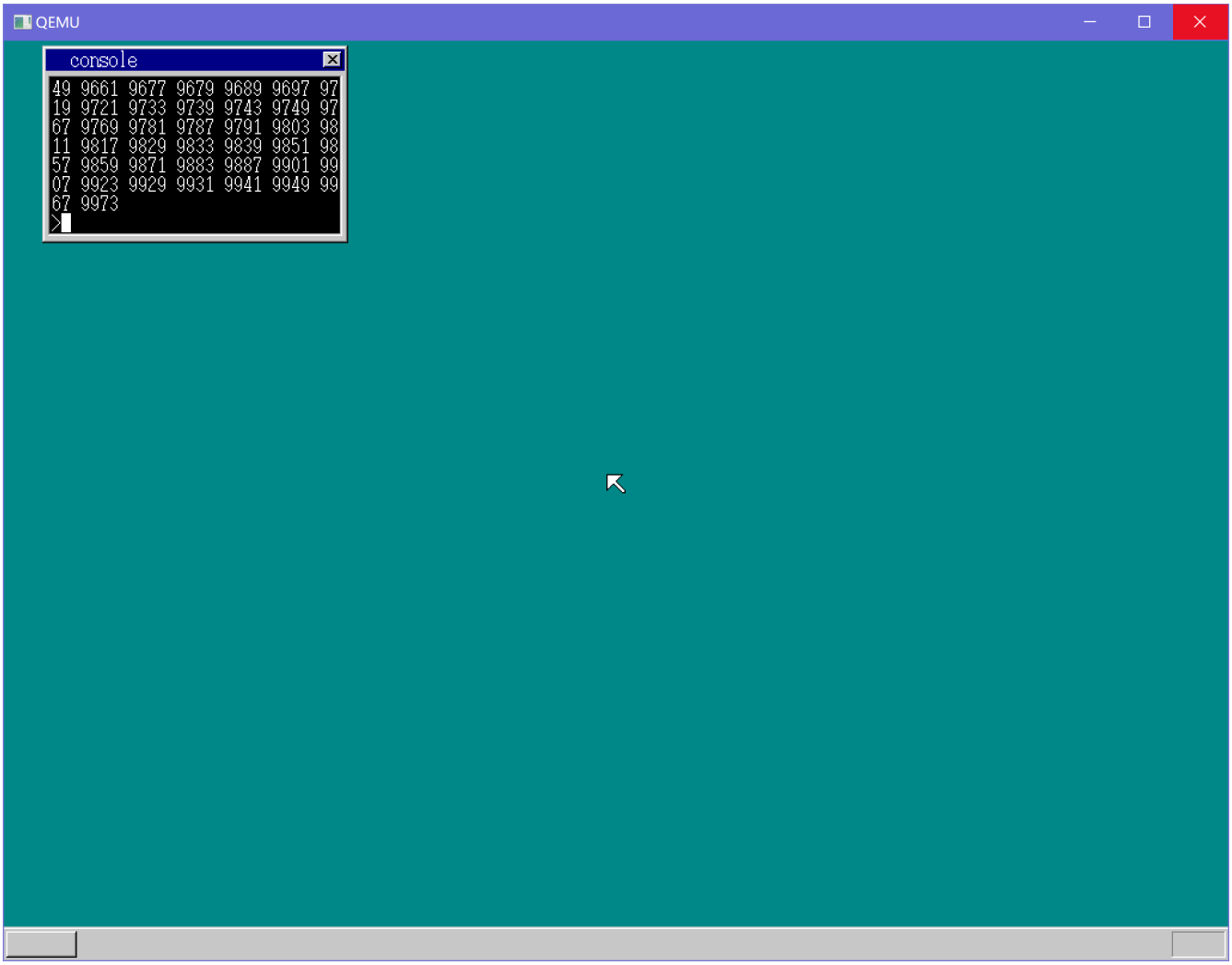
```
SUB ESP,EAX
JMP DWORD [ESP+EAX] ; 代替RET
```

POP EIP相当于`mov eip,[esp]`和`add esp,4` esp的值会出现误差

```
SUB ESP,EAX
JMP DWORD [ESP+EAX] ; 代替RET
ADD ESP,4
```

add又执行不到

```
SUB ESP,EAX
ADD ESP,4
JMP DWORD [ESP+EAX-4]
```



完成了__alloca之后，我们可以修改之前的winhelo和winhelo2了，将数组声明放到HariMain当中（之前这样做是会报错的）

接下来实现文件操作功能

打开	open
定位	seek
读取	read
写入	write
关闭	close

打开文件

EDX	21
EBX	文件名
EAX	文件句柄（为0时表示打开失败）（由操作系统返回）

关闭文件

EDX	22
EAX	文件句柄

文件定位

EDX	23
EAX	文件句柄
ECX	定位模式
	0：定位的起点为文件开头
	1：定位的起点为当前的访问位置
	2：定位的起点为文件末尾
EBX	定位偏移量

获取文件大小

EDX	24
EAX	文件句柄
ECX	文件大小获取模式
	0：普通文件大小
	1：当前读取位置从文件开头起算的偏移量
	2：当前读取位置从文件末尾起算的偏移量
EAX	文件大小（由操作系统返回）

文件读取

EDX	25
EAX	文件句柄
EBX	缓冲区地址
ECX	最大读取字节数
EAX	本次读取到的字节数（由操作系统返回）

定义数据类型

```
struct TASK {
    int sel, flags; /* sel为GDT编号*/
    int level, priority;
    struct FIFO32 fifo;
    struct TSS32 tss;
    struct SEGMENT_DESCRIPTOR ldt[2];
    struct CONSOLE *cons;
    int ds_base, cons_stack;
    struct FILEHANDLE *fhandle;
    int *fat;
};
struct FILEHANDLE {
    char *buf;
    int size;
    int pos;
};
```

在程序退出部分加入文件句柄释放部分

```
for (i = 0; i < 8; i++) {
    if (task->fhandle[i].buf != 0) {
        memman_free_4k(memman, (int) task->fhandle[i].buf, task->fhandle[i].size);
        task->fhandle[i].buf = 0;
    }
}
```

api

```
struct FILEHANDLE fhandle[8];
//.................
} else if (edx == 21) {
    for (i = 0; i < 8; i++) {
        if (task->fhandle[i].buf == 0) {
            break;
        }
```

```
    }
    fh = &task->fhandle[i];
    reg[7] = 0;
    if (i < 8) {
        finfo = file_search((char *) ebx + ds_base, (struct FILEINFO *) (ADR_DISKIMG +
0x002600), 224);
        if (finfo != 0) {
            reg[7] = (int) fh;
            fh->buf = (char *) memman_alloc_4k(memman, finfo->size);
            fh->size = finfo->size;
            fh->pos = 0;
            file_loadfile(finfo->clustno, finfo->size, fh->buf, task->fat, (char *)
(ADR_DISKIMG + 0x003e00));
        }
    }
} else if (edx == 22) {
    fh = (struct FILEHANDLE *) eax;
    memman_free_4k(memman, (int) fh->buf, fh->size);
    fh->buf = 0;
} else if (edx == 23) {
    fh = (struct FILEHANDLE *) eax;
    if (ecx == 0) {
        fh->pos = ebx;
    } else if (ecx == 1) {
        fh->pos += ebx;
    } else if (ecx == 2) {
        fh->pos = fh->size + ebx;
    }
    if (fh->pos < 0) {
        fh->pos = 0;
    }
    if (fh->pos > fh->size) {
        fh->pos = fh->size;
    }
} else if (edx == 24) {
    fh = (struct FILEHANDLE *) eax;
    if (ecx == 0) {
        reg[7] = fh->size;
    } else if (ecx == 1) {
        reg[7] = fh->pos;
    } else if (ecx == 2) {
        reg[7] = fh->pos - fh->size;
    }
} else if (edx == 25) {
    fh = (struct FILEHANDLE *) eax;
    for (i = 0; i < ecx; i++) {
        if (fh->pos == fh->size) {
            break;
        }
        *((char *) ebx + ds_base + i) = fh->buf[fh->pos];
        fh->pos++;
    }
    reg[7] = i;
```

```
}
```

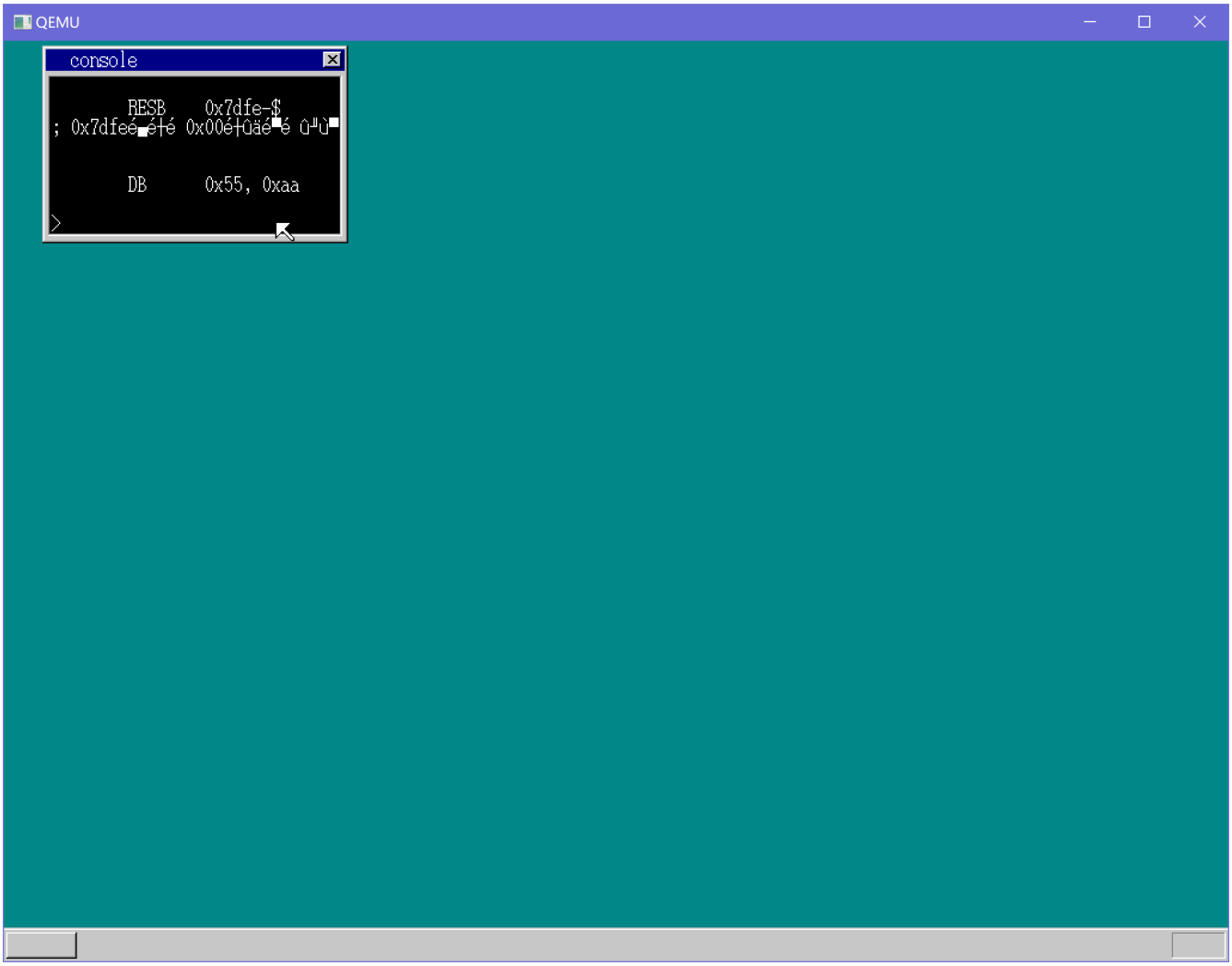
然后是apilib（就合一起写了）

```
_api_fopen: ; int api_fopen(char *fname);
        PUSH EBX
        MOV EDX,21
        MOV EBX,[ESP+8] ; fname
        INT 0x40
        POP EBX
        RET
_api_fclose: ; void api_fclose(int fhandle);
        MOV EDX,22
        MOV EAX,[ESP+4] ; fhandle
        INT 0x40
        RET
_api_fseek: ; void api_fseek(int fhandle, int offset, int mode);
        PUSH EBX
        MOV EDX,23
        MOV EAX,[ESP+8] ; fhandle
        MOV ECX,[ESP+16] ; mode
        MOV EBX,[ESP+12] ; offset
        INT 0x40
        POP EBX
        RET
_api_fsize: ; int api_fsize(int fhandle, int mode);
        MOV EDX,24
        MOV EAX,[ESP+4] ; fhandle
        MOV ECX,[ESP+8] ; mode
        INT 0x40
        RET
_api_fread: ; int api_fread(char *buf, int maxsize, int fhandle);
        PUSH EBX
        MOV EDX,25
        MOV EAX,[ESP+16] ; fhandle
        MOV ECX,[ESP+12] ; maxsize
        MOV EBX,[ESP+8] ; buf
        INT 0x40
        POP EBX
        RET
```

编写测试程序

```
/*typeipl.c*/
#include "apilib.h"
void HariMain(void)
{
    int fh;
    char c;
    fh = api_fopen("ipl10.nas");
    if (fh != 0) {
        for (;;) {
```

```
            if (api_fread(&c, 1, fh) == 0) {
                break;
            }
            api_putchar(c);
        }
    }
    api_end();
}
```



为了实现能够用应用程序打印任意文件的内容，我们还要实现读取命令行参数的功能

api设计

EDX	26
EBX	存放命令行内容的地址
ECX	最多可存放多少字节
EAX	实际存放了多少字节（由操作系统返回）

在TASK结构体中添加一个指针，cmdline，保存命令行内容

```
void console_task(struct SHEET *sheet, int memtotal)
{
    //..............
    task->cons = &cons;
    task->cmdline = cmdline;
    //..............
}
```

```
int *hrb_api(int edi, int esi, int ebp, int esp, int ebx, int edx, int ecx, int eax)
{
    //..............
    } else if (edx == 26) {
        i = 0;
        for (;;) {
            *((char *) ebx + ds_base + i) = task->cmdline[i];
            if (task->cmdline[i] == 0) {
                break;
            }
            if (i >= ecx) {
                break;
            }
            i++;
        }
        reg[7] = i;
    }
    return 0;
}
```

```
_api_cmdline: ; int api_cmdline(char *buf, int maxsize);
        PUSH EBX
        MOV EDX,26
        MOV ECX,[ESP+12] ; maxsize
        MOV EBX,[ESP+8] ; buf
        INT 0x40
        POP EBX
        RET
```

然后我们只要对cmdline进行处理，就可以获取参数了

```
#include "apilib.h"
void HariMain(void)
{
    int fh;
    char c, cmdline[30], *p;
    api_cmdline(cmdline, 30);
    for (p = cmdline; *p > ' '; p++) { }
    for (; *p == ' '; p++) { } /*跳过文件名和空格*/
    fh = api_fopen(p);
    if (fh != 0) {
```

```
        for (;;) {
            if (api_fread(&c, 1, fh) == 0) {
                break;
            }
            api_putchar(c);
        }
    } else {
        api_putstr0("File not found.\n");
    }
    api_end();
}
```

接下来引入全角字符支持。gb2312编码按区位的方式对字符进行索引。位是最小单位，然后是区

在GB2312中，字符编码的分类如下：

- 01区～09区：非汉字
- 10区～15区：空白
- 16区～55区：一级汉字
- 56区～87区：二级汉字
- 88区～94区：空白

接下来我们了解一下字库：

GB2312字库文件是HZK16，网搜下载一个。

HZK16字库是符合GB2312标准的16×16点阵字库,HZK16的GB2312-80支持的汉字有6763个，符号682个。其中一级汉字有3755个，按声序排列，二级汉字有3008个，按偏旁部首排列。

我们将压缩后的字库存入img中，然后再在系统启动后读取解压字库

```
nihongo = (unsigned char *) memman_alloc_4k(memman, 0x5d5d*32);
```

```
finfo = file_search("HZK16.fnt", (struct FILEINFO *) (ADR_DISKIMG + 0x002600), 224);
```

把汉字的langmode定为3.

注意HZK编码不同于sjis的地方在于，HZK是分上下而非左右。

修改graphic.c中的putfonts8_asc

```
if (task->langmode == 3) {
    for (; *s != 0x00; s++) {
        if (task->langbyte1 == 0) {
            if (0xa1 <= *s && *s <= 0xfe) {
                task->langbyte1 = *s;
            } else {
```

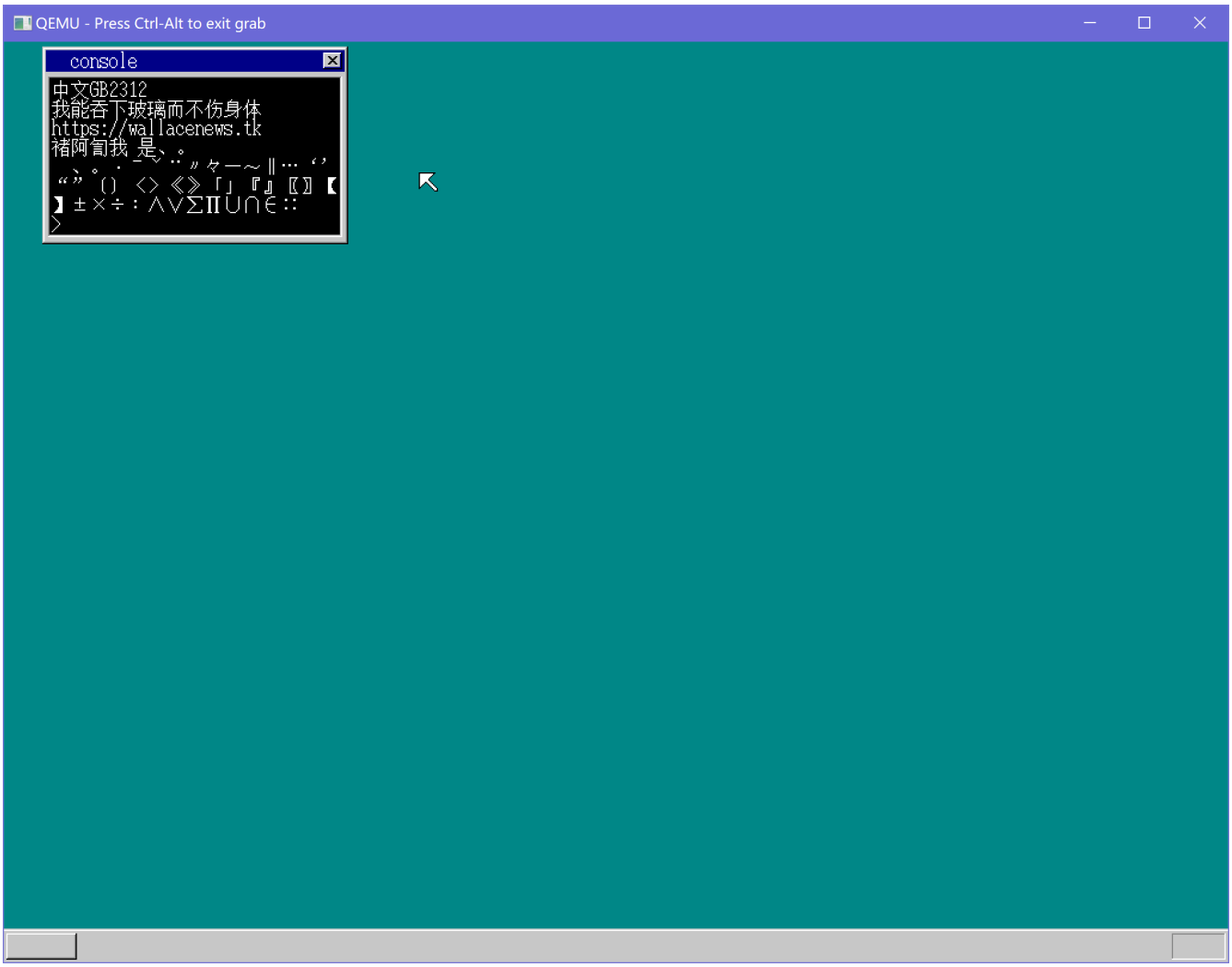
```
                putfont8(vram, xsize, x, y, c, hankaku + *s * 16);//只要是半角就使用hankaku里面的字符
            }
        } else {
            k = task->langbyte1 - 0xa1;
            t = *s - 0xa1;
            task->langbyte1 = 0;
            font = nihongo + (k * 94 + t) * 32;
            putfont32(vram,xsize,x-8,y,c,font,font+16);
        }
        x += 8;
    }
}
```

```
/*chklang.c*/
#include "apilib.h"

void HariMain(void)
{
    int langmode = api_getlang();
    static char s1[23] = {
        0x93, 0xfa, 0x96, 0x7b, 0x8c, 0xea, 0x83, 0x56, 0x83, 0x74, 0x83, 0x67,
        0x4a, 0x49, 0x53, 0x83, 0x82, 0x81, 0x5b, 0x83, 0x68, 0x0a, 0x00
    };
    static char s2[17] = {
        0xc6, 0xfc, 0xcb, 0xdc, 0xb8, 0xec, 0x45, 0x55, 0x43, 0xa5, 0xe2, 0xa1,
        0xbc, 0xa5, 0xc9, 0x0a, 0x00
    };
    static char s3[20] = {
        0xce, 0xd2, 0x20, 0xca, 0xc7, 0xa1, 0xa2, 0xa1, 0xa3, 0x0a, 0x00
    };
    int i;char j;
    if (langmode == 0) {
        api_putstr0("English ASCII mode\n");
    }
    if (langmode == 1) {
        api_putstr0(s1);
    }
    if (langmode == 2) {
        api_putstr0(s2);
    }
    if (langmode == 3) {
        api_putstr0("中文GB2312\n我能吞下玻璃而不伤身体\n");
        api_putstr0("https://wallacenews.tk\n");
        api_putstr0("褚阿訇");
        api_putstr0(s3);
        for(i=0xa1;i<0xcc;i++)
        {
            s3[0]=0xa1;
            j=i;
            s3[1]=j;
```

```
            s3[2]=0x00;
            api_putstr0(s3);
        }
    }
    api_end();
}
```

试一下吧



大功告成！我们支持中文了！