河南工業大學

课程设计报告

一个小型图形界面操作系统

课程名称：操作系统原理

专业班级：软件 1305 班

姓　　名：田劲锋

学　　号：201316920311

指导教师：刘扬

完成时间：2015 年 6 月 25 日

软件工程 专业课程设计任务书

学生姓名	田劲锋	专业班级	软件 1305 班	学 号	201316920311
题 目	一个小型图形界面操作系统				
课题性质	其他		课题来源	自拟课题	
指导教师	刘扬		同组姓名	无	
主要内容	操作系统是控制应用程序执行的程序，并充当应用程序和计算机硬件之间的接口。一个操作系统的主要功能有： 1. 处理器管理 2. 存储器管理 3. 设备管理 4. 文件管理 现在的桌面操作系统多是多任务分时操作系统。				
任务要求	目标是完成一个基本可用的图形界面操作系统，包括如下基本模块： 1. 进程：中断处理、多任务调度、系统保护 2. 存储管理：内存分配、进程空间管理 3. I/O 系统：鼠标、键盘和屏幕的控制 4. 文件系统：文件与可执行程序的读取和加载 系统提供命令行用户接口和图形化用户接口，允许使用 C 语言编写系统应用程序，可以从 FAT12 格式软盘启动。				
参考文献	川合秀实. 30天自制操作系统. 人民邮电出版社, 2012 W. Stallings. 操作系统: 精髓与设计原理（第6版）. 机械工业出版社, 2010 R. E. Bryant, 等. 深入理解计算机系统系统（第2版）. 机械工业出版社, 2010 A.S.Tanenbaum, 等. 操作系统设计与实现. 电子工业出版社, 2007 W. R. Stevens, 等. UNIX环境高级编程（第3版）. 人民邮电出版社, 2014				
审查意见	指导教师签字： 教研室主任签字：　　2015 年 6 月 25 日				

说明：本表由指导教师填写，由教研室主任审核后下达给选题学生，装订在设计（论文）首页

目录

参考文献

[1] W. Stallings. 操作系统: 精髓与设计原理. 6 edn. 机械工业出版社, 2010

[2] R. E. Bryant, D. R. O'Hallaron. 深入理解计算机系统系统. 2 edn. 机械工业出版社, 2010

[3] W. R. Stevens, S. A. Rago. UNIX环境高级编程. 3 edn. 人民邮电出版社, 2014

[4] 川合秀实. 30天自制操作系统. 人民邮电出版社, 2012

[5] A. S. Tanenbaum, A. S. Woodhull. 操作系统设计与实现. 电子工业出版社, 2007

[6] 邓建松, 彭冉冉, 陈长松. LaTeX 2ε科技排版指南. 北京: 科学出版社, 2001

[7] B. W. Kernighan, D. M. Ritchie, (Editors) C 程序设计语言. 2 edn. 北京: 机械工业出版社, 2004

[8] D. E. Knuth. The Art Of Computer Programming. Pearson Education, 1968–2011

[9] 高德纳. 计算机程序设计艺术. 北京: 国防工业出版社, 1992–2010

A 程序清单

```
.
├── Makefile
├── Makefile.rule
├── apilib.h
└── app
    ├── Makefile
    ├── Makefile.rule
    ├── a
    │   ├── Makefile
    │   └── a.c
    ├── app_make.txt
    ├── beepdown
    │   ├── Makefile
    │   └── beepdown.c
    ├── cat
    │   ├── Makefile
    │   └── cat.c
    ├── color
    │   ├── Makefile
    │   └── color.c
    ├── color2
    │   ├── Makefile
    │   └── color2.c
    ├── hello3
    │   ├── Makefile
    │   └── hello3.c
    ├── hello4
    │   ├── Makefile
    │   └── hello4.c
    ├── hello5
    │   ├── Makefile
    │   └── hello5.nas
    ├── lines
    │   ├── Makefile
    │   └── lines.c
    ├── noodle
    │   ├── Makefile
    │   └── noodle.c
    ├── primes
    │   ├── Makefile
    │   └── primes.c
    ├── primes2
    │   ├── Makefile
    │   └── primes2.c
    ├── primes3
    │   ├── Makefile
    │   └── primes3.c
    ├── star1
    │   ├── Makefile
    │   └── star1.c
    ├── stars
    │   ├── Makefile
    │   └── stars.c
    ├── stars2
    │   ├── Makefile
    │   └── stars2.c
    ├── walk
    │   ├── Makefile
    │   └── walk.c
    └── winhelo
        ├── Makefile
        └── winhelo.c
```

```
    ├── winhelo2
    │   ├── Makefile
    │   └── winhelo2.c
    └── winhelo3
        ├── Makefile
        └── winhelo3.c
├── lib
    ├── Makefile
    ├── alloca.nas
    ├── api001.nas
    ├── api002.nas
    ├── api003.nas
    ├── api004.nas
    ├── api005.nas
    ├── api006.nas
    ├── api007.nas
    ├── api008.nas
    ├── api009.nas
    ├── api010.nas
    ├── api011.nas
    ├── api012.nas
    ├── api013.nas
    ├── api014.nas
    ├── api015.nas
    ├── api016.nas
    ├── api017.nas
    ├── api018.nas
    ├── api019.nas
    ├── api020.nas
    ├── api021.nas
    ├── api022.nas
    ├── api023.nas
    ├── api024.nas
    ├── api025.nas
    └── api026.nas
└── sys
    ├── Makefile
    ├── ZpixEX2-12.fnt
    ├── asmhead.nas
    ├── bootpack.c
    ├── bootpack.h
    ├── console.c
    ├── dsctbl.c
    ├── fifo.c
    ├── file.c
    ├── graphic.c
    ├── int.c
    ├── ipl10.nas
    ├── keyboard.c
    ├── memory.c
    ├── mouse.c
    ├── mtask.c
    ├── naskfunc.nas
    ├── sheet.c
    ├── timer.c
    ├── unifont-7.0.06.hex
    └── window.c

23 directories, 95 files
```

Listing 1: sys/bootpack.h

```
#ifndef BOOTPACK_H
#define BOOTPACK_H

#define SYSNAME "PriPara OS"
#define SYSVERS "28"
#define SYSNAMEVER SYSNAME " " SYSVERS

/* asmhead.nas */
typedef struct BOOTINFO { /* 0x0ff0-0x0fff */
    char cyls; /* 启动区读盘终止处 */
    char leds; /* 键盘灯状态 */
    char vmode; /* 显卡模式 */
    char reserve;
    short scrnx, scrny; /* 分辨率 */
    char* vram;
} bootinfo_t;

#define ADR_BOOTINFO 0x00000ff0
#define ADR_DISKIMG 0x00100000

/* naskfunc.nas */
void io_hlt(void);
void io_cli(void);
void io_sti(void);
void io_stihlt(void);
int io_in8(int port);
void io_out8(int port, int data);
int io_load_eflags(void);
void io_store_eflags(int eflags);
void load_gdtr(int limit, int addr);
void load_idtr(int limit, int addr);
int load_cr0(void);
void store_cr0(int cr0);
void load_tr(int tr);
void asm_inthandler0c(void);
void asm_inthandler0d(void);
void asm_inthandler20(void);
void asm_inthandler21(void);
void asm_inthandler27(void);
void asm_inthandler2c(void);
unsigned int memtest_sub(unsigned int start, unsigned int end);
void farjmp(int eip, int cs);
void farcall(int eip, int cs);
void asm_hrb_api(void);
void start_app(int eip, int cs, int esp, int ds, int* tss_esp0);

/* fifo.c */
typedef struct FIFO32 {
    int* buf;
    int p, q, size, free, flags;
    struct TASK* task;
} fifo32;

void fifo32_init(fifo32* q, int size, int* buf, struct TASK*);
int fifo32_put(fifo32* fifo, int data);
int fifo32_get(fifo32* fifo);
int fifo32_status(fifo32* fifo);

/* graphic.c */
void init_palette(void);
void set_palette(int start, int end, unsigned char* rgb);
void boxfill8(unsigned char* vram, int X, unsigned char c,
    int x0, int y0, int x1, int y1);
void boxsize8(unsigned char* vram, int X, unsigned char c,
    int x0, int y0, int width, int height);
void init_screen8(char* vram, int x, int y);
void putfont8(char* vram, int xsize, int x, int y, char c, char* font);
void putfonts8_asc(char* vram, int xsize, int x, int y, char c, unsigned char* s);
void init_mouse_cursor8(char* mouse, char bc);
void putblock8_8(char* vram, int vxsize, int pxsize,
    int pysize, int px0, int py0, char* buf, int bxsize);

```

```
/* 16位色 */
#define base03 0
#define base02 1
#define base01 2
#define base00 3
#define base0 4
#define base1 5
#define base2 6
#define base3 7
#define yellow 8
#define orange 9
#define red 10
#define magenta 11
#define violet 12
#define blue 13
#define cyan 14
#define green 15

#define BGM cyan

/* 字体 */
#define FNT_H 12
#define FNT_W 6 // FNT_H / 2
#define FNT_OFFSET 726 // 60 * FNT_H

/* 鼠标指针 */
#define CURSOR_X 12
#define CURSOR_Y 19

/* dsctbl.c */
typedef struct SEGMENT_DESCRIPTOR {
    short limit_low, base_low;
    char base_mid, access_right;
    char limit_high, base_high;
} segment_descriptor;

typedef struct GATE_DESCRIPTOR {
    short offset_low, selector;
    char dw_count, access_right;
    short offset_high;
} gate_descriptor;

void init_gdtidt(void);
void set_segmdesc(segment_descriptor* sd, unsigned int limit, int base, int ar);
void set_gatedesc(gate_descriptor* gd, int offset, int selector, int ar);

#define ADR_IDT 0x0026f800
#define LIMIT_IDT 0x000007ff
#define ADR_GDT 0x00270000
#define LIMIT_GDT 0x0000ffff
#define ADR_BOTPAK 0x00280000
#define LIMIT_BOTPAK 0x0007ffff
#define AR_DATA32_RW 0x4092
#define AR_CODE32_ER 0x409a
#define AR_LDT 0x0082
#define AR_TSS32 0x0089
#define AR_INTGATE32 0x008e

/* int.c */
struct KEYBUF {
    unsigned char data[32];
    int next_r, next_w, len;
};

void init_pic(void);
void inthandler21(int* esp);
void inthandler27(int* esp);
void inthandler2c(int* esp);

#define PIC0_ICW1 0x0020
#define PIC0_OCW2 0x0020
#define PIC0_IMR 0x0021
#define PIC0_ICW2 0x0021
#define PIC0_ICW3 0x0021
#define PIC0_ICW4 0x0021
#define PIC1_ICW1 0x00a0
#define PIC1_OCW2 0x00a0
#define PIC1_IMR 0x00a1
#define PIC1_ICW2 0x00a1
#define PIC1_ICW3 0x00a1
#define PIC1_ICW4 0x00a1

/* keyboard.c */
void inthandler21(int* esp);
void wait_KBC_sendready(void);
void init_keyboard(fifo32* fifo, int data0);

#define PORT_KEYDAT 0x0060
#define PORT_KEYCMD 0x0064

/* mouse.c */
typedef struct MOUSE_DEC {
    unsigned char buf[3], phase;
    int x, y, btn;
} mouse_dec;

void inthandler2c(int* esp);
void enable_mouse(fifo32* fifo, int data0, mouse_dec* mdec);
int mouse_decode(mouse_dec* mdec, unsigned char dat);

/* memory.c */
#define MEMMAN_FREES 4090 // 大约是32KB
#define MEMMAN_ADDR 0x003c0000

typedef struct FREEINFO { /* 空闲块 */
```

```
    unsigned int addr, size;
} freeinfo_t;

typedef struct MEMMAN { /* 内存管理 */
    int frees, maxfrees, lostsize, losts;
    freeinfo_t free[MEMMAN_FREES];
} memman_t;

unsigned int memtest(unsigned int start, unsigned int end);
void memman_init(memman_t* man);
unsigned int memman_total(memman_t* man);
unsigned int memman_alloc(memman_t* man, unsigned int size);
int memman_free(memman_t* man, unsigned int addr, unsigned int size);
unsigned int memman_alloc_4k(memman_t* man, unsigned int size);
int memman_free_4k(memman_t* man, unsigned int addr, unsigned int size);

/* sheet.c */
#define MAX_SHEETS 256

typedef struct SHEET {
    unsigned char* buf;
    int bxsize, bysize, vx0, vy0, alpha, height, flags;
    struct SHTCTL* ctl;
    struct TASK* task;
} sheet_t;

typedef struct SHTCTL {
    unsigned char *vram, *map;
    int xsize, ysize, top;
    sheet_t* sheets[MAX_SHEETS];
    sheet_t sheets0[MAX_SHEETS];
} shtctl_t;

shtctl_t* shtctl_init(memman_t* memman, unsigned char* vram, int xsize, int ysize);
struct SHEET* sheet_alloc(shtctl_t* ctl);
void sheet_setbuf(sheet_t* sht, unsigned char* buf, int xsize, int ysize, int col_inv);
void sheet_updown(sheet_t* sht, int height);
void sheet_refresh(sheet_t* sht, int bx0, int by0, int bx1, int by1);
void sheet_slide(sheet_t* sht, int vx0, int vy0);
void sheet_free(sheet_t* sht);

/* timer.c */
#define MAX_TIMER 512

typedef struct TIMER {
    struct TIMER* next;
    unsigned int timeout;
    char flags, flags2;
    fifo32* fifo;
    int data;
} timer_t;

typedef struct TIMERCTL {
    unsigned int count, next, using;
    timer_t *t0, timers0[MAX_TIMER];
} timerctl_t;

extern timerctl_t timerctl;

void init_pit(void);
timer_t* timer_alloc(void);
void timer_free(timer_t* timer);
void timer_init(timer_t* timer, fifo32* fifo, int data);
void timer_settime(timer_t* timer, unsigned int timeout);
void inthandler20(int* esp);
int timer_cancel(timer_t* timer);
void timer_cancelall(fifo32* fifo);

/* mtask.c */
#define MAX_TASKS 1024 /* 最大任务数 */
#define TASK_GDT0 3 /* 从GDT的哪里开始分配TSS */
#define MAX_TASKS_LV 100
#define MAX_TASKLEVELS 10

typedef struct TSS32 {
    int backlink, esp0, ss0, esp1, ss1, esp2, ss2, cr3;
    int eip, eflags, eax, ecx, edx, ebx, esp, ebp, esi, edi;
    int es, cs, ss, ds, fs, gs;
    int ldtr, iomap;
} tss32;

typedef struct TASK {
    int sel, flags; /* sel存放GDT的编号 */
    int level, priority; /* 优先级 */
    fifo32 fifo;
    tss32 tss;
    struct SEGMENT_DESCRIPTOR ldt[2];
    struct CONSOLE* cons;
    int ds_base;
    int cons_stack;
    struct FILEHANDLE *fhandle;
    int *fat;
    char *cmdline;
} task_t;

typedef struct TASKLEVEL {
    int running; /* 运行中任务数 */
    int now; /* 当前运行中任务 */
    task_t* tasks[MAX_TASKS_LV];
} tasklevel_t;

typedef struct TASKCTL {
    int now_lv; /* 活动中的等级 */
    char lv_change; /* 下次是否改变等级 */
    tasklevel_t level[MAX_TASKLEVELS];
```

```
    task_t tasks0[MAX_TASKS];
} taskctl_t;

extern timer_t* task_timer;
extern taskctl_t* taskctl;

task_t* task_now(void);
task_t* task_init(memman_t* memman);
task_t* task_alloc(void);
void task_run(task_t* task, int level, int priority);
void task_switch(void);
void task_sleep(task_t* task);

/* window.c */
#define WIN_TOP 28
#define WIN_LEFT 8

void make_window8(unsigned char* buf, int xsize, int ysize, char* title, char act);
void make_wtitle8(unsigned char* buf, int xsize, char* title, char act);
void make_textbox8(sheet_t* sht, int x0, int y0, int sx, int sy, int c);
void putfonts8_asc_sht(sheet_t* sht, int x, int y, int c, int b, char* s, int l);
void change_wtitle8(sheet_t* sht, char act);

/* console.c */
#define CONS_COLN 80 /* 列数（自定义） */
#define CONS_LINE 30 /* 行数（自定义） */
#define CONS_COLW (FNT_W * CONS_COLN) /* 列宽 */
#define CONS_LINH (FNT_H * CONS_LINE) /* 行高 */
#define CONS_LEFT 3 /* 左边宽度 */
#define CONS_TOP 23 /* 顶部高度 */
#define CONS_WINW (CONS_COLW + CONS_LEFT * 2) /* 窗口宽度 */
#define CONS_WINH (CONS_LINH + CONS_TOP + CONS_LEFT) /* 窗口高度 */

typedef struct CONSOLE {
    sheet_t* sht;
    int cur_x, cur_y, cur_c;
    timer_t* timer;
} console;
typedef struct FILEHANDLE {
    char *buf;
    int size;
    int pos;
} filehandle;
void console_task(sheet_t* sheet, unsigned int memtotal);
void cons_putchar(console* cons, int chr, char move);
void cons_newline(console* cons);
void cons_putstr0(console* cons, char* s);
void cons_putstr1(console* cons, char* s, int l);
void cons_runcmd(char* cmdline, console* cons, int* fat, unsigned int memtotal);
void cmd_mem(console* cons, unsigned int memtotal);
void cmd_cls(console* cons);
void cmd_dir(console* cons);
void cmd_exit(console* cons, int* fat);
void cmd_start(console* cons, char* cmdline, int memtotal);
void cmd_open(console* cons, char* cmdline, int memtotal);
int cmd_app(console* cons, int* fat, char* cmdline);
int* hrb_api(int edi, int esi, int ebp, int esp, int ebx, int edx, int ecx, int eax);
int* inthandler0c(int* esp);
int* inthandler0d(int* esp);
void asm_end_app(void);
void hrb_api_linewin(sheet_t* sht, int x0, int y0, int x1, int y1, int col);

/* file.c */
typedef struct FILEINFO {
    unsigned char name[8], ext[3], type;
    char reserve[10];
    unsigned short time, date, clustno;
    unsigned int size;
} fileinfo;

void file_readfat(int* fat, unsigned char* img);
void file_loadfile(int clustno, int size, char* buf, int* fat, char* img);
fileinfo* file_search(char* name, fileinfo* finfo, int max);

/* bootpack.c */
task_t* open_constask(sheet_t* sht, unsigned int memtotal);
sheet_t* open_console(shtctl_t* shtctl, unsigned int memtotal);

#endif
```