Outline of Operating System

My OS will include these features as follows:

1. The OS will support multiple threads, because the thread is a lightweight process which could execute a specific function and use shared memory. For the mechanisms, I want to define a library (e.g., thread_create(), thread_exit()) to implement a lifecycle of threads, and I will design a Thread Control Block (TCB) to monitor the status of them.
2. The OS will support events, because it must provide the communication and interaction between controller and screen. For example, if a user click a botton on the controller, then it will trigger the event which links the API provide by OS and the botton. Finally, the corresponding result will be shown on the screen. I will design a Event Queue which stores evnets and handles with them one by one. Additionally, for the registration of handlers, I will implement a table to build the links between handlers and events. So, if an event is trigged, the OS will use this table to find the address of the corresponding handles.
3. The OS will provide virtual address and TLB for the management of memory.
4. The OS will provide the concurrency mechanisms, such as the input from the controller, the vision of the screen, and the tolerance of system errors.
5. The user input will trigger the events which provide the interation between users and OS, then OS will call APIs for the game process. For the first version of OS, I do not think it can support a second controller.
6. In order to simplify the game development, I think the underlying OS must provide APIs to developers for the interaction of the video graphics. So, if a game developer calls the APIs provided by OS, the video graphics will be changed.
7. The OS will prevent the rogue cartridges from damaging the console by two ways: 1) we can release a certificate for the cartridges, that is, the rogue cartridges which do not have certificates cannot be run on the OS; 2) we can define two modes (user mode and kernel mode) in this OS. In user mode, process cannnot execute privileged instructions which might hurts the OS. And, all cartridges will be run under the user mode.

List of APIs

Kernel Mode

Name

event_handler - a handler which will be used to handler with events. If an event happens, this API will be triggered, and then redirects the event to the corresponding function

Signature

#include "event.h"

void *event_handler(struct signal sig, void *handler, void *args)

Parameters

struct signal sig - a data structure which defines differnt signal and each signal presents a kind of event

void *handler - the function or handler which will be called by this event

void *args - the arguments of the handler

Return Values

None

Name

boot_OS - system booting

Signature

#include <os.h>

int boot_OS() - system booting

Parameters

None

Return Values

It will return 0 if successfully, and the failure will return -1

Name

terminate_OS - terminates the OS in a gentle way

Signature

#include <os.h>

void terminate_OS()

Parameters

None

Return Values

None

User Mode

Name

load_cartridge - loads the cartridge process, and runs the corresponding game

Signature

#include <cartridge.h>

int load_cartridge(char *filename)

Parameters

char *filename - the cartridge's name

Return Values

It will return 0 if successfully, and the failure will return -1

Name

terminate_cartridge - if a rogue cartridge is running or the cartridge cannot be run , OS call it to terminate the cartridge process.

Signature

#include <cartridge.h>

Void terminate_cartridge()

Parameters

None

Return Values

None

Threads

Name

thread_create - creates a thread to execute a more specific task

Signature

#include <thread.h>

int thread_create(struct thread *thr, void *func, void *arg)

Parameters

struct thread *thr - a data structure to control and manage a thread

void *func - a function the thread executes

void *arg - the arguments of the function

Return Values

It will return 0 if successfully, and the failure will return -1

Name

thread_start - starts a thread

Signature

#include <thread.h>

int thread_start(struct thread *thr)

Parameters

struct thread *thr - a data structure to control and manage a thread

Return Values

It will return 0 if successfully, and the failure will return -1

Name

thread_stop - stops a thread

Signature

#include <thread.h>

int thread_stop(struct thread *thr)

Parameters

struct thread *thr - a data structure to control and manage a thread

Return Values

It will return 0 if successfully, and the failure will return -1

Name

thread_exit - terminates a thread and releases its resources

Signature

#include <thread.h>

int thread_exit(struct thread *thr)

Parameters

struct thread *thr - a data structure to control and manage a thread

Return Values

It will return 0 if successfully, and the failure will return -1

Controller

Name

direction_controller - the direction of movement on the screen

Signature

#include <controller.h>

void *direction_controller(int direction, struct time duration)

Parameters

int direction - 0: up, 1: down, 2: left, 3 right

struct time duration - how much time the user presses on the button

Return Values

None

Name

cmd_controller - generates an interrupt via the chipset

Signature

#include <controller.h>

void *cmd_controller()

Parameters

None

Return Values

None

Name

function_controller - define the functionality of four toggle buttons

Signature

#include <controller.h>

void *function_controller(int behavior)

Parameters

int behavior - different behavior represents different toggle buttons

Return Values

None

Text Mode

Name

generate_text - generates characters or text

Signature

#include <text.h>

int generate_text(char *text)

Parameters

char *text - the content of text

Return Values

It will return 0 if successfully, and the failure will return -1

Name

update_text - updates characters or text

Signature

#include <text.h>

int update_text(char *text)

Parameters

char *text - the content of text

Return Values

It will return 0 if successfully, and the failure will return -1

Name

remove_text - removes characters or text

Signature

#include <text.h>

int remove_text()

Parameters

None

Return Values

It will return 0 if successfully, and the failure will return -1

Graphics Mode

Name

generate_palette - generates a global array which contains 16777216 possible colors

Signature

#include <graphics.h>

void generate_palette(struct palette *pal)

Parameters

struct palette *pal - a data structure which define and manage the palette

Return Values

None

Name

destroy_palette - destorys the palette array

Signature

#include <graphics.h>

void destroy_palette(struct palette *pal)

Parameters

struct palette *pal - a data structure which define and manage the palette

Return Values

None

Name

load_backgroud - loads the background

Signature

#include <graphics.h>

int load_background(char *filename, struct background *bg)

Parameters

char *filename - the name of background images

struct background *bg - the parameters of background

Return Values

It will return 0 if successfully, and the failure will return -1

Name

update_backgroud - changes the background

Signature

#include <graphics.h>

int update_backgroud(char *filename, struct background *bg)

Parameters

char *filename - the name of background images

struct background *bg - the parameters of background

Return Values

It will return 0 if successfully, and the failure will return -1

Name

generate_image - generates an image wihch contains sprites

Signature

#include <graphics.h>

int generate_image(struct image *img)

Parameters

struct image *img - a data structure which contains the information of the image

Return Values

It will return 0 if successfully, and the failure will return -1

Name

refresh_images - updates the image

Signature

#include <graphics.h>

int refresh_images(struct image *img)

Parameters

struct image *img - a data structure which contains the information of the image

Return Values

It will return 0 if successfully, and the failure will return -1

Name

destroy_images - destroys the image and releases the resources

Signature

#include <graphics.h>

int destroy_images(struct image *img)

Parameters

struct image *img - a data structure which contains the information of the image

Return Values

It will return 0 if successfully, and the failure will return -1

Memory

Name

xmalloc - allocates memory

Signature

#include <memory.h>

void xmalloc(int size)

Parameters

int size - the size of memory

Return Values

None