CMER

Centre for Mobile Education and Research

BlackBerry Operating System

Week II

Overview

- Introduction
- Multitasking
- Multithreading
- Messaging Services
  - Asynchronous
  - Synchronous
- File System Services
  - Random Access to Files
  - Reading from the File System
- BlackBerry Operating System APIs

Outline (Cont.)

- Radio Communication APIs
- System APIs
- System Devices Events
- Keypad APIs
- LCD APIs
- Other SDKs

Introduction

- BlackBerry OS has a multitasking environment.
- It enables heavy use of input devices like trackball, and scroll wheel. It does not support touchpad.
- It is an event-driven Operating System.
- Later BlackBerry Smartphone's CPU architecture is based on ARM XScale. The other BlackBerry devices has Intel-based processors.

Introduction (Cont.)

- It supports multitasking and multithreading applications.
- Security: Any application that want to use certain BlackBerry functionality must be digitally signed.

Multitasking Feature

- **BlackBerry OS employs co-operative multitasking so no application can preempt another application in mid-stream, unless that application explicitly yields control.**
- **All applications run simultaneously and are managed by an application server.**

Multitasking Feature (Cont.)

- Each application at startup, receives an execution thread. They may also create and destroy additional thread dynamically.
- At no time preemption can occur between applications. Such a design removes the need for mutual exclusion mechanisms and semaphores.
- However, the task will take several seconds.

Multitasking Feature (Cont.)

- Threads can be run in the background waiting for messages or data for processing, or they can be run in foreground having control of user interfaces, display, and input like keystrokes .
- The foreground tasks include displaying context such as display bitmap, show on the LCD, and receiving all keypad and trackwheel inputs.

Multitasking Feature (Cont.)

- Each task that is not in the foreground, still maintains a copy of the LCD display bitmap in its display context. So, it can manipulate it at any time.
- When the foreground thread is changed to a different thread or task, the new task's display bitmap is placed on the LCD and now, it receives all inputs.

Threads

- **Whenever, the foreground is switched from one application to other application, the new foreground application receives a message of type *SWITCH_FOREGROUND* and wisely, the previous foreground application receives a message of type *SWITCH_BACKGROUND*.**

- **Foreground threads can be switched by applications requesting a new application to be replaced on the foreground by calling *RimRequestForeground***

Multitasking Feature (Cont.)

- Communication between OS and threads is done by a messaging system.
- Like event-driven systems, applications receive messages describing system events and associate parameters. Then, they post them to threads to process.

Message Services

- As described earlier, BB OS is an event-driven OS.
- This means that BB applications receive all external notifications through events sent to the applications.
- Applications process the events. Since the process is completed, they call the RIMGETMESSAGE function to receive the next event.

Message Services (Cont.)

- If there is no event, the application blocks the send process, allowing other applications to run.
- If also other applications do not have any event to process, the application puts the CPU in a standby state until the next event.
- There are two ways of sending messages between two tasks: synchronous and asynchronous.

Message Services: Asynchronous

- **For asynchronous communication (non-blocking send), applications send message to another application messages' queue by calling *RimPostMessage*. The destination receives messages. The sending process continues execution immediately after the call to *RimPostMessage*.**

Message Services: Synchronous

- In Synchronous communications (blocking send), applications send messages to other application's message queue by calling *RimSendMessage* and blocks the sending process until it receives responses from the destination.
- The destination application receives the messages by calling RimGetMessage, put it in the queue and processes all earlier events in its message queue.

Message Services: Synchronous

- Then, it processes the message, and any result of the processed message is sent to the location specified by the sending process (if applicable).
- The sending application is unblocked and it returns form RimSendMessage.
- Thus, in order to avoid deadlock from occuring, when you write an application, you should run the synchronous sends in background threads.

File System Services

- File System APIs are used to access the device's persistent memory.
- The BB device file system mechanism is different from traditional file systems.
- All the file's system non-violate data is stored in flash memory.
- Both read and write operations are done on a flash space.

File System Services (Cont.)

- Writing a single word on the flash involves the following tasks[1]:
  - Saving the content of entire enclosing flash sector
  - Erasing the entire flash sector
  - Rewriting the entire contents of the flash sector with the changed content

File System Services (Cont.)

- The process of rewriting is like a log-structured file system like follows:
  - All writes are performed sequentially at the end of the log file
  - When data that already exists in the log file is modified, it is added to the end of the log. The old copy of data left in the log but it marks as dirty.
  - When file system runs out of space, old segments of the log (with a dirty flag) are cleaned.

Random Access to Files

- In desktops, file systems use indexing structures to access files.
- This can not be used in handheld devices because of limited resources.
- As a result, random access to a file is done by sequentially scanning the file from the beginning. This method in handheld device is fast, because the entire file system in a flash memory.

Reading from the File System

- In order to optimize the speed of reading from the file system, the file system provides a form read-only memory mapped access.
- In disk-based file systems, memory mapped file access is implemented through the memory management system.
- In handhelds, file system explicitly provides lookup tables for all of the records in the fule system.

BlackBerry OS APIs

- Radio Communication API
- Serial Communication API
- File System API
- Keypad API
- LCD API
- System API

Radio Communication APIs

- The BB Radio communication APIs provide easy access to radio network for sending and receiving data.
- To know about these API, please refer to Radio Developer's Guide.
- Regarding the Radio Communication, the following events may occur:

Radio Communication APIs

- MESSAGE_RECEIVED
- MESSAGE_SENT
- MESSAGE_NOT_SENT
- SIGNAL_LEVEL
- NETWORK_STARTED
- BASE_STATION_CHANGE
- RADIO_TURNED_OFF
- MESSAGE_STATUS

System APIs

- System APIs include a variety of functionalities. They provide functions for thread management, message handling, and task switching.
- They also provides function for memory allocation, timers, alarm and the system clock.
- Threads can provide information about battery status, language, handheld status.

System Devices Events

- The following system events may occur:
  - SWITCH_BACKGROUND
  - SWITCH_FOREGROUND
  - POWER_OFF/POWER_UP
  - BATTERY_LOW
  - TASK_LIST_CHANGED
  - Timer Device Events
  - RTC device Events

Keypad APIs

- Keypad functions are :
  - KeypadBeep: turns key tones on or off.
  - KeypadRate: configure the keypad for auto key repeat.
  - KeypadRegister: allows an application to intercept global hot keys.
- The events include:
  - THUMB_CLICK, THUMB_UNCLICK, THUMB_ROLL_UP, THUMB_ROLL_DOWN
  - KEY_DOWN, KEY_REPEAT, KEY_STATUS

LCD APIs

- LCD APIs provides a variety of functions for LCD.

Other APIs

- There are a variety of APIs for BlackBerry OS including:
  - Radio API SDK
  - UI Engine API SDK
  - Database API SDK
  - Extended API including:
    - Address Book API SDK.
    - AutoText API SDK.
    - Messaging API SDK.
    - Ribbon API SDK.
- In order to access them, please refer to SDKs in the RIM Web site.

References

- [1] BlackBerry Operating System APIs: Developer's Guide.
- [2] BlackBerry Software Development Kit.