Software Architectures

I/O Interfaces and Service

José Costa

Software for Embedded Systems

Departamento de Engenharia Informática (DEI)
Instituto Superior Técnico

2015-10-05

Outline

- Software Architectures
- Round-Robin
- Round-Robin with Interrupts
- Function-Queue Scheduling
- Multitasking Operating System (Real-Time OS)

- How much control you need to have over system response
  - Absolut response time requirements
  - Speed of your processor
- Very few response time requirements $\Rightarrow$ very simple architecture
- Rapid response and various processing requirements $\Rightarrow$ more complex architecture

Characteristics of Software Architectures

- Priorities
- Worst response time for task code
- Stability of response when the code changes
- Simplicity

Main Software Architectures

- Round-Robin
- Round-Robin with Interrupts
- Function-Queue Scheduling
- Multitasking Operating System (Real-Time OS)

Round-Robin

- Simplest imaginable architecture
- No interrupts
- Main loop checks each I/O device in turn
  - Services any that need service
- No shared data
- Suitable for systems with no latency concerns

Round-Robin - Adequate for some jobs

- Small number of I/O devices
- No particular lenghty processing
- No tight response requirements

Its simplicity makes it a very attractive choice for some systems

Round-Robin - Example

```
void main (void) {
  while (TRUE) {
    if (I/O Device A needs service) {
      Take care of I/O Device A
      Handle data to/from I/O Device A
    }
    if (I/O Device B needs service) {
      Take care of I/O Device B
      Handle data to/from I/O Device B
    }
...
    if (I/O Device Z needs service) {
      Take care of I/O Device Z
      Handle data to/from I/O Device Z
    }
  }
}
```

Round-Robin

Disadvantages

- If any one device needs response in less time than it takes the microprocessor to get around the main loop in the worst-case scenario, then the system won't work
  - a workaround is to service the device more than once in the loop
- Even if none of the required response times are absolute deadlines, the system may not work well if there is any lengthy processing to do.
- A single additional device or requirement may break everything

Round Robin Examples

- Digital multimeter
  - Measure: resistance, current and potential
  - For each situation select the right scale
- Washing machine
- Airbag controller
- Anything that displays information
- ...

Main Software Architectures

- Round-Robin
- Round-Robin with Interrupts
- Function-Queue Scheduling
- Multitasking Operating System (Real-Time OS)

Round-Robin with Interrupts

- Somewhat more sofisticated architecture
- Interrupt rotines deal with the very urgent needs of the hardware and then sets flags
- The main loop polls the flags and does any follow-up processing required by the interrupts

Round-Robin with Interrupts

Example (1/2)

```
BOOL fDeviceA = FALSE;
BOOL fDeviceB = FALSE;
...
BOOL fDeviceZ = FALSE;

void interrupt vHandleDeviceA (void) {
  Take care of Device A
  fDeviceA = TRUE;
}
void interrupt vHandleDeviceB (void) {
  Take care of Device B
  fDeviceB = TRUE;
}
...
void interrupt vHandleDeviceZ (void) {
  Take care of Device Z
  fDeviceZ = TRUE;
}
```

Round-Robin with Interrupts

Example (2/2)

```
void main (void) {
  while (TRUE) {
    if (fDeviceA) {
      fDeviceA = FALSE;
      Handle data to/from I/O Device A
    }
    if (fDeviceB) {
      fDeviceB = FALSE;
      Handle data to/from I/O Device B
    }
...
    if (fDeviceZ) {
      fDeviceZ = FALSE;
      Handle data to/from I/O Device Z
    }
  }
}
```

Round-Robin with Interrupts

Advantages

- More control over priorities

- Interrupt rotines can get good response
  - All the processing that you put in the interrupt rotines has a higher priority than the task code in the main rotine

Round-Robin with Interrupts

Disadvantages

- Not as simple as plain round-robin
- All of the task code executes at the same priority
  - One solution is to move the task code into the interrupt rotine
  - But this may not be acceptable
  - Or we can interlace the main loop tests of the flags
- Worst-case is when the interrupt for a given device happens just after the round-robin loop passes the task code for that device, and every other device needs service

Round-Robin with Interrupts - Examples

- Stopwatches
- Modern washing machines
- Coffee machines
- Microwave ovens
- Central heating units
- Traffic light controllers
- Other: data-bridge devices and barcode scanners

Main Software Architectures

- Round-Robin
- Round-Robin with Interrupts
- Function-Queue Scheduling
- Multitasking Operating System (Real-Time OS)

Function-Queue Scheduling

- Interrupt routines adds function pointers to a queue

- The main routine just reads the pointers from the queue and calls the functions

Function-Queue Scheduling

Functions calling

- Main does not have to call the functions in the order that the interrupt routines occurred
- It can call them based on any priority scheme that suits your purposes
- Any task code functions that need quicker response can be executed earlier
- All it takes is a little clever coding in the routines that queue up the function pointers

Function-Queue Scheduling

Example (1/2)

```
Queue of function pointers;

void interrupt vHandleDeviceA (void)
{
  Take care of Device A
  Put function_A on queue of function pointers
}
void interrupt vHandleDeviceB (void)
{
  Take care of Device B
  Put function_B on queue of function pointers
}
...
void interrupt vHandleDeviceZ (void)
{
  Take care of Device Z
  Put function_Z on queue of function pointers
}
```

Function-Queue Scheduling

Example (2/2)

```
void main(void) {
  while (TRUE) {
    while (Queue of function pointers is empty);
    Call first function on queue
  }
}

void function_A (void) {
  Handle actions required by Device A
}
void function_B (void) {
  Handle actions required by Device B
}
...
void function_Z (void) {
  Handle actions required by Device Z
}
```

Function-Queue Scheduling

- Worst wait for the highest-priority task code function is the length of the longest of the task code functions
- Better response than the round-robin-with-interrupts response
- To workaround worst wait problem you can rewrite long functions in pieces
  - each of which schedules the next piece by adding it to the function queue

Function-Queue Scheduling

Disadvantages

- More complicated than previous architectures
- Response for lower-priority task code functions may get worse
  - they may never run

Main Software Architectures

- Round-Robin
- Round-Robin with Interrupts
- Function-Queue Scheduling
- Multitasking Operating System (Real-Time OS)

Multitasking Operating System

Real-Time OS

- Interrupt routines take care of the most urgent operations
- Then signal that there is work for the task to do
- Needs a Real-Time Operating System

- RTOS handles the necessary signaling between the interrupt routines and the task code
  - no need for shared variables
- No loop in our code decides what needs to be done next
  - RTOS knows about the various task-code subroutines and will run whichever of them is more urgent at any given time
- RTOS can suspend one task code subroutine in the middle of its processing in order to run another

Multitasking Operating System

Real-Time OS

```
void interrupt vHandleDeviceA (void) {
  Take care of Device A and Set signal X
}
void interrupt vHandleDeviceB (void) {
  Take care of Device B and Set signal Y
}
void Task1 (void) {
  while (TRUE) {
    Wait for Signal X
    Handle data to/from Device A
  }
}
void Task2 (void) {
  while (TRUE) {
    Wait for Signal Y
    Handle data to/from Device B
  }
}
```

Multitasking Operating System

Real-Time OS

- The worst-case wait for the highest-priority task code is zero
- System's response will be relatively stable, even when you change the code
- Widely available and you do not have to write the code yourself
- The main disadvantage is that it uses a certain amount of processing time

Selecting an Architecture

- Select the simplest architecture that will meet response requirements
- If your system has response requirements that might necessitate using a RTOS, you should lean toward using a RTOS
- You can create hybrids of the architectures

Outline

- Software Architectures
- Round-Robin
- Round-Robin with Interrupts
- Function-Queue Scheduling
- Multitasking Operating System (Real-Time OS)

References

David E. Simon. An Embedded Software Primer. Addison-Wesley, 1999. Ch. 5.

Next class

- Software architectures - exercises