Programming for Embedded System

Overview of ANSI C

- Although C is a popular language , Software developed using C compilers of different vendors led to portability problems.
- To achieve portability, American National Standard Institute(ANSI) released the ANSI C specification in 1989.
- International Standard Organization(ISO) adapted this standard as C99.

Programming for Embedded System

C99 Features

- Function Prototyping
- Parameter passing
- Standard Include Files (assert, ctype, errno, float, limits, locale, math, setjmp, signal, stdarg, stdef, stdio, string, stdlib are the standard include files).

- ISO developed extensions to C99 standard specifically for embedded software development.
- Standard is called ‘Extensions for the Programming Language C to support embedded processors’.
- The latest version of this standard is ISO/IEC 9899:1999.Majority of components available to the market conform to C99.

Programming for Embedded System

The 3 important aspects pertaining to the extension are listed below.

- **New Data Types**
  - Most of the embedded software involves only fixed point arithmetic.
  - New Data type are “fract” and “accum” .
    - “fract” can take the values between -1.0 and +1.0 & “accum” will have both integer and fraction part.
  - <stdfix.h> defines the macros that specify the precision of the fixed point types and declares functions that support fixed point arithmetic.
  - data types also support boolean data type and complex numbers.

Programming for Embedded System

- **Address Space definition**
  - Embedded Systems will have different memory chips(RAM, ROM, Flash), function calls are provided to handle multiple address spaces.
  - OS and Compilers will hide these differences through these function calls.
  - Function calls provide a uniform mechanism for defining an address space as well as variables within a specified address space.
- **I/O hardware addressing**
  - Generally , the software to address I/O devices is not portable.
  - File <iohw.h> contains the access functions to promote portability of I/O hardware driver source code across different platforms.
  - 'iohw' access functions create platform independent interfaces between I/O driver source code and underlying access method used when addressing the I/O registers in a given platform.

Programming for Embedded System

- Bit Manipulation using C
  - Programming tasks that require bit manipulation include low-level device control, error detection and correction algorithms, data compression, encryption algorithms, and optimization.
  - Source code that does bit manipulation makes use of the bitwise operations: AND, OR, XOR, NOT, and bit shifts.

Programming for Embedded System

- Bit Manipulation using C
  - Bitwise Operator
    - AND (&)

bit a	bit b	a & b (a AND b)
0	0	0
0	1	0
1	0	0
1	1	1

    - **OR (|)**

bit a	bit b	a | b (a OR b)
0	0	0
0	1	1
1	0	1
1	1	1(0 with carry 1)

Programming for Embedded System

- Bit Manipulation using C
  - Bitwise Operator
    - XOR (^)

bit a	bit b	a ^ b (a XOR b)
0	0	0
0	1	1
1	0	1
1	1	0

bit a	~a (complement of a)
0	1
1	0

Programming for Embedded System

- Bit Manipulation using C
  - Bitwise Operator
    - Left Shift (<<)
      - If variable “a” contains the bit pattern 11100101, then a<<1 will be 11001010.
    - Right Shift (>>)
      - If variable “a” contains the bit pattern 11100101, then a>>1 will be 01110010.
    - For both shifter, Blank spaces generated are filled up by zeroes.
- Some more operator discussed in program (Setting, Clearing, Toggle)

Programming for Embedded System

- Bit Manipulation using C
- Some more operator discussed in program
  - Setting: Use bitwise OR operator to set a bit
    - `n |= (1 << K) where K is the bit that is to be set`
  - Clearing: Use bitwise AND operator to clear a bit
    - `n &= ~(1 << k) where k is the bit that is to be cleared`
  - Toggle: Use bitwise EX-OR operator to toggle a bit
    - `n ^= (1 << k) where k is the bit that is to be cleared`
  - Examples in program

Programming for Embedded System

- Calculation of CRC
  - •Cyclic Redundancy Check(CRC) calculations are required in data communication software for error detecting.
  - •When we transmit a bit stream , an additional bits are added to provide error detection capability.
  - •The bit stream is divided with a polynomial and the result of the division is the CRC.
  - •So, when we transmit data CRC is appended to the original data and data is sent over medium.
  - •At receiver end, CRC is again calculated and calculated CRC is matched with the received CRC.
  - •If they matched the data is received correctly other wise retransmission of data is requested.

Programming for Embedded System

- Memory Management
  - In embedded system software development , we need to allocate memory.
  - The memory allocation programming is must if we are writing our own operating system kernel.
  - The function calls **malloc()** and **free()** are used to allocate memory and free the memory.

Programming for Embedded System

- Timing of Programs
  - It is important to measure the time taken to execute a portion of a program for embedded software development.
  - It is useful to carry out simulation studies of algorithms to be implemented on the embedded system.
  - The timing analysis gives the valuable information as to which portions of the code are taking longer time, so that these portion can be written in assembly language, if required, to improve the speed.
  - In C language, the function time() and clock() are used are used to measure the time taken by the program segments.

Programming for Embedded System

- Device Drivers
  - Device driver are low level interfaces to control the system hardware.
  - The system calls open, read, write, close, are used to access the device drivers.
  - In Unix/ Linux, each device is considered a file and each file will have file descriptor. The standard file descriptor are
    - 0 for stdin (the keyboard)
    - 1 for stdout (the monitor)
    - 2 for stderr (the monitor)

Programming for Embedded System

- Device Drivers
  - Device drivers categories
    - Character device (Serial Port).
    - Block device (Hard-disk).
  - Device driver software is not portable. A separate device driver is required for each operating system. But things are changing recently which will result in portability of the device driver software.
  - We need to know the system calls of each operating system to write the device driver.
  - We need to write the details of the hardware for writing the device driver.

Programming for Embedded System

- Device Drivers
  - Device driver wizard are available through which we can easily build the device drivers.
  - Most of the operating system vendors also provide the Device Driver Kits (DDKs) to facilitate easy development.
  - Some of the device driver wizards which help in fast device driver development are :
    - QNX's DDK for graphics, input, printer, networking cards , and USB.

Programming for Embedded System

- Productivity Tools
  - The productivity tools are very important for developing software systematically. Some of the important tools are listed below.
    - **Makefile**
      - When different number of people are working on a project, each person will work on a module and then all the modules are combined in single software package.
      - Even, if some person is working alone , he may like to write code by splitting into different files.
      - Makefile is an excellent utility to combine different project files and create the executable files.

Programming for Embedded System

- Productivity Tools
  - Debugger
    - It is a methodical process of finding and reducing the bugs (defects) in computer program. The utility ‘gdb’ is the mostly used debugger for embedded system.
  - Profiler
    - While developing embedded software on the host system, we need to check the amount of time it takes to execute some of the functions/modules.
    - In Unix operating system, the profiler command ‘gproof’ is used to find the execution time taken by different function.
    - The timing information will help to find the timing-consuming functions so that we can optimize the code or we can change some portion of code in assembly language for faster execution.

Programming for Embedded System

- Productivity Tools
  - **Indenting**
    - When we write the lengthy programs, indenting the code increases the readability and hence maintainability of the code.
    - To indent in a C program we use ‘indent’ shell command.
    - Some of the indent option available are discussed below.
      - -bad (Blank lines after indentation).
      - - sob(swallow optional blank) Remove unnecessary blank lines in the source code).
  - **Revision Control**
    - Since Software Engineers develop different versions of a program, it is good practice to record new changes that have been made so that that the proper information is always present currently or for future references.
    - Source Code Control System (SCCS) and Revision Control System (RCS) are used extensively in Unix/ Linux.

Programming for Embedded System

- Code Optimization
  - In embedded software development , the two important considerations are speed of execution and memory requirement for the software.
  - Writing code in assembly language increases speed of execution and also results in optimal code.
  - However maintaining the code written in assembly language is difficult , hence code are written high level language.
  - Some portion of the code which take longer execution time are written in assembly language.
  - Also while writing code in high level language, Code optimization can be done to make the execution faster.

Programming for Embedded System

- Code Optimization Guidelines
  - Eliminate Dead Code.
    - Avoiding, declaration and Initialization of variable that will not be used at all.
    - Function that will not be called at all.
    - Profilers help in elimination of dead code.
  - Remove Unnecessary debugging code.
    - Lots of code is written during development time for debugging process.
    - Unnecessary code should be removed from the final software.
  - Avoid Recursion.
    - Recursive function require a large stack ,resulting in higher memory requirements.
    - The best way to avoid recursion is to use lookup- table that contains pre calculated information.

Programming for Embedded System

- Code Optimization Guidelines
  - Avoid floating-point operations.
    - If necessary, floating-point operations can be converted into fixed point operations. (For example, the value 1.23 can be represented as 1230 in a fixed-point data type with scaling factor of 1/1000, and the value 1,230,000 can be represented as 1230 with a scaling factor of 1000).
  - Use unsigned integers.
    - Use unsigned int instead of int if you know the value will never be negative. Some processors can handle unsigned integer arithmetic considerably faster than signed int.

Programming for Embedded System

- Code Optimization Guidelines
  - Remove loop-invariant code.
    - The code that has no effect on the loop counter is called loop invariant code.

      In C program :

```
for (i=0;i<10;i++)
{
    x=1.0/b; % Loop Invariant Code.
    y=x+1;
}
```

  - Use the correct optimization level of the compiler.
    - The compiler has an in-built optimizer. Generally the compiler provide different optimization level. The higher level will have higher optimization. Some bugs appear only on higher optimization level but on this optimization level, the compilation time is more. So, based on the need, the optimization level can be set.

Programming for Embedded System

- Although programming in assembly language will lead to increase in the speed of execution, but maintaining the code written in assembly language is difficult.
- That's why High Level Language are also preferred to write code for embedded system.
- In high level language , we can also choose between structured programming language and object oriented programming language.
- While writing with OOP programming language, the overhead of language is slightly higher in terms of memory requirements. But, OOP language facilitate reuse.
- However, due to reduced cost of memory device, OOP languages(C++ & Java) are becoming very important languages for embedded software development.
- In all, the choice of programming language for embedded system design depends upon the project and programmer skills.

Programming for Embedded System

- **Java**
  - It is platform independent, so we can run the application on any Java virtual machine (JVM) regardless of computer architecture.
  - Presently, Java is one of the most preferred language for the development of application in mobile device.
  - Java language is likely to play a prominent role even for embedded software development with the development of small footprint Java Virtual Machines.
  - The application developed in Java is converted into bytecode using a Java Compiler. The bytecode is independent of the platform. If Java Virtual Machine (JVM) is running on a machine, the JVM will interpret the bytecode and produce the output.

Programming for Embedded System

- To facilitate development of applications for different market segments. Sun Microsystems released 3 edition of Java.
  - Java 2 Enterprise Edition (J2EE). (For Server Market)
  - Java 2 Standard Edition (J2SE). (For Desktop Market)
  - Java 2 Micro Edition (J2ME). (For consumer appliances and embedded devices market)

Programming for Embedded System

- **Java 2 Micro Edition (J2ME).**
  - J2ME addresses two types of configurations based on the capabilities of the devices.
    - Connected Device Configuration (CDC).
      - These devices can be connected to high speed data networks.
      - They run TCP/IP protocol stack and have high processing power and large memory in the range of 2 to 16 MB.
      - Developing application for these devices is similar to developing applications for desktop computers.
      - Example : Internet TV.

Programming for Embedded System

- Connection, Limited Device Configuration (CLDC).
  - These devices are connected to low speed networks such as the wireless networks.
  - These device do not run TCP/IP protocol stack.
  - They have low processing power and have small memory capacity of the order of 128 KB and 512 KB.
  - To facilitate development of applications that can run on these devices, CLDC devices contain Kilobytes Virtual Machine (KVM) which requires very little memory – 40 to 80 KB of memory.
  - To run Java applications on CLDC devices, the total memory requirement is about 128 KB for KVM, the class libraries and the application.
  - For large applications, the memory requirement is about 256 KB, out of which 40 to 80 KB is for KVM and the rest is for class libraries and application.

Programming for Embedded System

- Connection, Limited Device Configuration (CLDC).
  - ClDC devices have 16-bit or 32-bit processors and 160 to 512 KB of memory.
  - 128 KB of non-volatile memory is required for KVM and CLDC libraries.
  - 32 KB of volatile memory is required for runtime and object memory.
  - For CLDC appliances, Mobile Information Device Profile (MIDP) has been defined to develop interoperable applications.

Programming for Embedded System

- Connection, Limited Device Configuration (CLDC).
  - The architecture of a Java-enabled CLDC device running MIDP is shown in below figure.

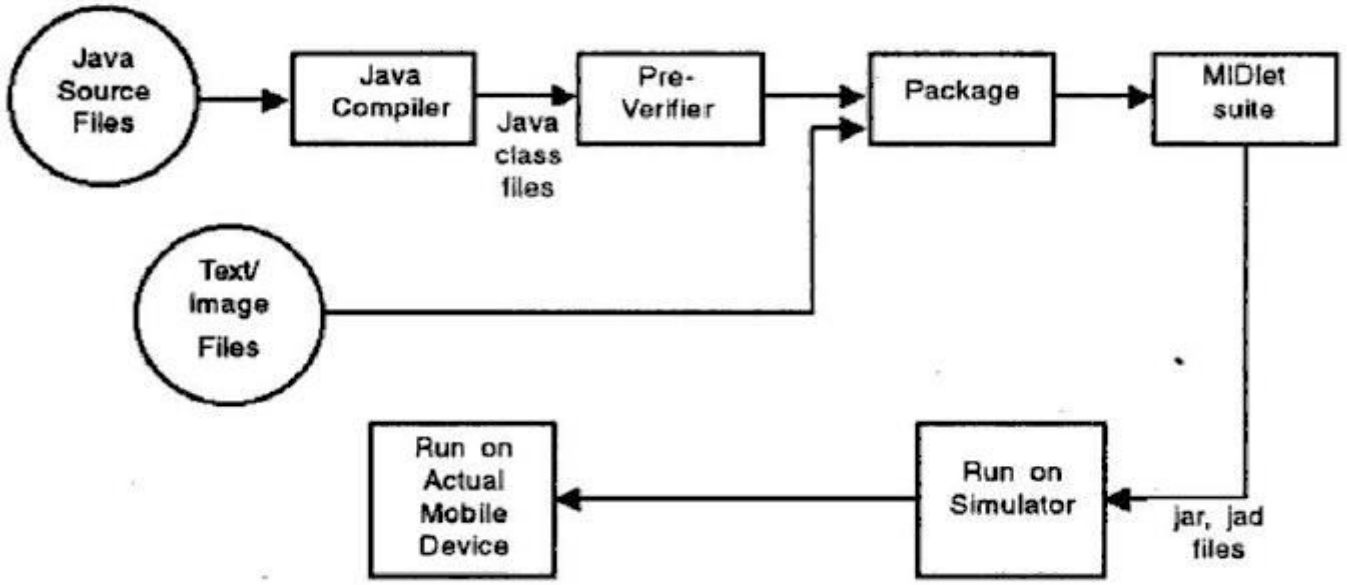

Fig. 3.3 MIDlet Development Process

Programming for Embedded System

•Connection, Limited Device Configuration (CLDC).

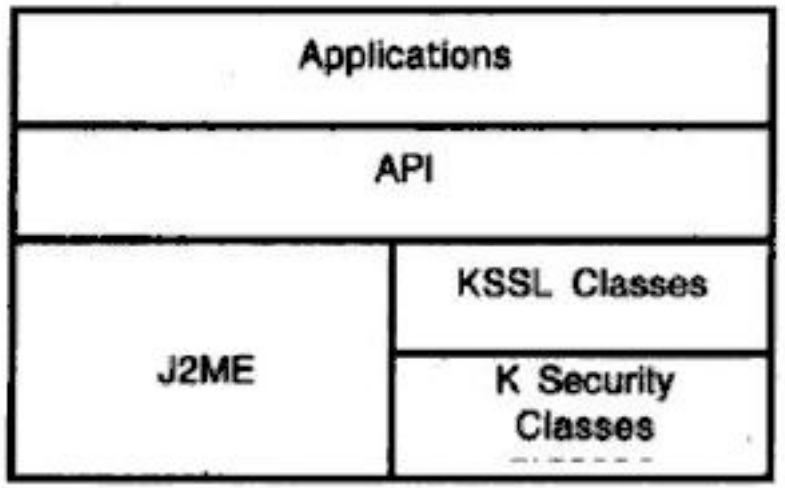

Fig. 3.4: Securing Applications for MIDP in J2ME

Programming for Embedded System

- MIDlets development process
  - The Java source files are compiled into .class files, which are pre-verified and converted into packages to form MIDlet suites.
  - The MIDlet suite is run on the emulator to test it's functionality and is subsequently ported to actual mobile device.
  - When a mobile device accesses the application on the internet security is an issue.
  - A set of K Security classes and KSSL classes are available for KVM with which the developer can develop secure applications.