Systems Programming

Meeting 1:
Introduction

Welcome

- Course Registration
  - Register to submit your data at http://teaching.yfolajimi.com/register.html
- Course details/resources:
  - http://teaching.yfolajimi.com/Sys-Prog-14-15.html
- Major Coverage
  - Unix/Linux, Assembly Language, C, C++
- Lecture Notes
  - To be provided in course website

Course Outline

- Introduction to Systems Programming
- Introduction to the Unix Operating System
- Linux Distributions & Installation
- Linux Vs Windows
- Terminal V/s File Manager
- Must Know Linux/Unix Commands
- File Permissions in Linux/Unix
- Introduction to Assembler and Machine Language
- Compiling, assembling, linking, loading
- Programming with C
- Differences between C & Java
- Operator Overloading and polymorphism
- Exception Handling
- Programming with C++

What is systems programming?

- The activity of computer programming system software.
- The primary distinguishing characteristic of systems programming when compared to application programming is that application programming aims to produce software which provides services to the user (e.g. word processor)
- systems programming aims to produce software which provides services to the computer hardware (e.g. disk defragmenter).
- It requires a greater degree of hardware awareness

Attributes that characterize systems programming:

- The programmer will make assumptions about the hardware and other properties of the system that the program runs on, and will often exploit those properties,
  - for example by using an algorithm that is known to be efficient when used with specific hardware.
- Usually a low-level programming language or programming language dialect is used that:
  - can operate in resource-constrained environments
  - is very efficient and has little runtime overhead
  - has a small runtime library, or none at all
  - allows for direct and "raw" control over memory access and control flow
  - lets the programmer write parts of the program directly in assembly language

Attributes that characterize systems programming:

- Often systems programs cannot be run in a debugger. Running the program in a simulated environment can sometimes be used to reduce this problem.
- In system programming, often limited programming facilities are available. Because of those limitations, monitoring and logging are often used; operating systems may have extremely elaborate logging subsystems.
  - The use of automatic garbage collection is not common
  - debugging is sometimes hard to do.
  - The runtime library, if available at all, is usually far less powerful, and does less error checking.
- Implementing certain parts in operating system and networking requires systems programming, for example implementing Paging (Virtual Memory) or a device driver for an operating system.

System programming language

- a programming language used for system programming; such languages are designed for writing system software, which usually requires different development approaches when compared to application software.

History

- Originally systems programmers invariably wrote in assembly language.
- Experiments with hardware support in high level languages in the late 1960s led to such languages as PL/S, BLISS, BCPL, and extended ALGOL for Burroughs large systems.
- In the 1980s, C became ubiquitous, aided by the growth of Unix.
- More recently C++ has seen some use, for instance a subset of it is used in the I/O Kit drivers of Mac OS X.[1]

System programming language (History)

- The earliest system software was written in assembly language for reasons including efficiency of object code, compilation time, and ease of debugging. Application languages such as FORTRAN were used for system programming, although they usually still required some routines to be written in assembly language.

System programming language (History)

- **Mid-level languages**
  - Mid-level languages "have much of the syntax and facilities of a higher level language, but also provide direct access in the language (as well as providing assembly language) to machine features.“
  - One of the earliest of these mid-level programming languages was PL360, which had the general syntax of ALGOL 60, but whose statements directly manipulated CPU registers and memory.
  - Other languages in this category are MOL-360 and PL/S.

System programming language (History)

- **Higher-level languages**
  - An early example of this kind of language is LRLTRAN,which extended Fortran with features for character and bit manipulation, pointers, and directly-addressed jump tables.
  - Subsequently, languages such as C were developed, where the combination of features was sufficient to write system software, and a compiler could be developed that generated efficient object programs on modest hardware
  - Although many other languages were developed, C and C++ are the ones that have survived.
  - System Programming Language (SPL) is also the name of a specific language on the HP 3000 computer series, used for its operating system HP Multi-Programming Executive, and other parts of its system software.

Major languages

Language	Originator	Date	Derivation	Used for
ESPOL	Burroughs Corporation	1961	Algol 60	MCP
PL/I	MIT	1964	-	Multics
PL360	Niklaus Wirth	1968	Algol 60	Algol W
C	Dennis Ritchie	1969	BCPL	Unix
PL/S	IBM	196x	PL/I	OS/360
BLISS	Carnegie Mellon University	1970	Algol-PL/I[5]	VMS (portions)
PL/8	IBM	197x	PL/I	AIX
PL-6	Honeywell, Inc.	197x	PL/I	CP-6
SYMPL	CDC	197x	JOVIAL	NOS subsystems, most compilers, FSE editor
C++	Bjarne Stroustrup	1979	C, Simula	See C++ Applications[6]
D	Digital Mars	2001	C++	XomB
Go	Google	2009	C, Pascal, CSP	Google minor systems[7]
Rust	Mozilla Research[8]	2012	C++, Haskell, Erlang, Ruby	Servo layout engine