LINUX AND ITS CHARACTERISTICS

1. Characteristics of Linux

Linux is

1. **Multiuser** - allows number of users to work with the system at the same time
2. **Multitasking** - supports true preemptive multitasking. All processes independently of each other.
3. **Multiprocessing** - from kernel Version2.0 onwards, linux supports multiprocessor architectures. Applications are distributed across several processors.
4. **Architecture Independent** - runs almost on all platform that are able to process bits and bytes. examples like IBM s390, Sparc, ARM, etc
5. **Demand Load Executables** - only those parts of a program actually required for execution are loaded into memory.
6. **Support POSIX1003.1 standard** - Linux since version. onwards supports POSIX 1003.1, which defines a minimum interface to a Unix like operating system
7. **Memory protected Mode** - uses the processor's memory protection mechanisms to provent the process from accessing memory allocated to the system kernel or other processes. This is for the security of the system
8. **Shared Libraries** - it is a collection of routines needed by a program to work. There are number of standard libraries used by more than one process at the same time. so these libraries are made to load into memory for once and all the programs can make use of it.
9. **Support Various Filesystem** - supports various file systems like Proc, Ext2, Ext3 and Ext4. The most commonly used file system is Ext3 and Ext4 also developed, both are journaling file systems.

2. Kernel Directory Structure in Linux

In Linux the Kernel source is available under **/usr/src/linux.**

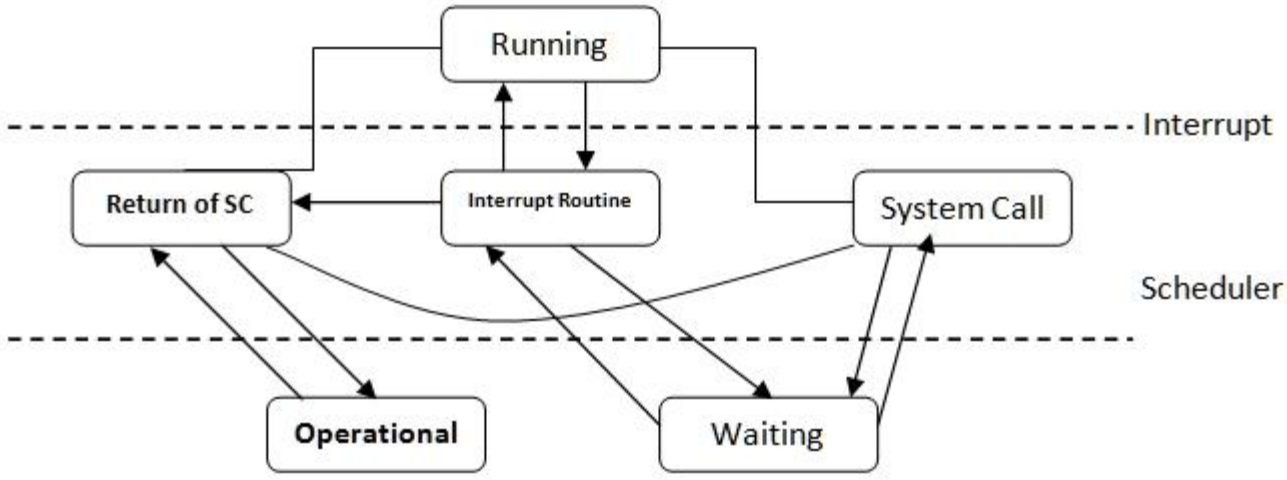
- Architecture dependent code is available in the following directory structure
    - **arch/alpha** – for the DEC Alpha Architecture
    - **arch/x86** – for the Intel 32 bit Architecture
    - **arch/arm** – for the ARM Architecture
    - **arch/ia64** – for the intel 64 bit architecture
    - **arch/m68k** – for the 68000 architecture and compatible processors.
- **init/** directory contains all the functions needed to start the kernel.
- **kernel/** – central sections of the kernel. Most important system calls are implemented here.
- **arch/x86/mm or mm/** – takes care of memory management by requesting and releasing kernel memories.
- **fs/** is the virtual file system interface. Some important file systems are proc, ext2,ext3,. The proc file systems is used for system management.
- **drivers/** – every operating system requires drivers for its hardware components. These are held in this directory and classified into groups according to their subdirectories like the following
    - **drivers/char** – character oriented devices
    - **drivers/block** – block oriented devices
    - **drivers/i2c** – a generic i2c driver

    - **drivers/scsi** – the SCSI interface
    - **drivers/usb** – drivers for the USB subsystem
- **ipc/** – indicates the Inter process communication
- **lib/** – indicates the standard C library functions

3. Introduction to Linux Kernel

- The Linux or the unix kernel is just developed under the microkernel architecture…
- The Microkernel provides only the necessary minimum of functionality (inter process communication and the memory management) and can be accordingly be implemented in a small and compact form. Building on this microkernel, the remaining functions of the operating system are relocated to the autonomous processes communicating with the microkernel via a well defied interface.
- Generally, microkernel systems have been created whose performance can be improved by the monolithic systems. Since linux provides slow i386 architecture, so linux is developed using the monolithic design..
- the code size of linux mainly occupied by the device drivers and similars. on the other hand, the central routines of process and memory management are relatively small and easily understood, with 13000 lines of C code in each
- Thus LINUX is successfully tries to make use of the advantages of a microkernel architecture without giving up its original monolithic design.

States of a Process



Running – The task is active and running inthe non privileged user mode. This state can be interrupted only when there is a system call or the interrupt.

Interrupt Routine – The interrupt routine becomes active when the hardware signals an exception condition which may be new characters from the keyboard or any other hardware interrupt.

System calls – system calls are initiated by software interrupts

Waiting – the process is waiting for an external event. E.g. waiting for a button press…

Ready – the process is ready to run under the CPU, but some other process is currently running under the cpu.

Return from system call – This state is adopted after the end of the system call or end of the interrupts.

Source : http://engineeringcourses.files.wordpress.com/2010/04/osp-lecture-notes1.pdf