Software Architecture

- Services Provided by an Operating System
- Categories of Embedded Systems
- Architecture of Embedded Operating Systems
- Application Software, Communication Software, Development/Testing Tools
- Communication Software

Software Architecture

- Based on capabilities, Operating System can be divided into different categories:

Embedded Operating System

- **Single-tasking OS versus Multi-tasking:**
  - In Single tasking OS, only one task is carried out at a time. For instance, if you are using a word processor, then it is necessary to quit this application before invoking another application.
  - In multitasking OS, multiple tasks can be run simultaneously. For example, listening music and surfing internet.
  - MS DOS is a single tasking operating system where as Windows and UNIX are multitasking operating system.

- **<u>Single-user OS versus Multi-user OS</u>**
  - Single User OS :Only one user can use the system at a time. (MS-DOS, Windows 9X).
  - Multi User OS : Multiple User can use the system simultaneously. (Windows and UNIX). In multi-tasking and multi user OS, system need to manage,
  - o **Each user logged on to the system and their workspace .**
  - o **Allocate resources to the jobs they want to run.**
  - o **Keep logs of how much processing time and resources they use .**
  - o **Maintain security.**

- **Command-driven OS versus GUI-based OS:**
  - Its all about user interface to access computer's resource.
  - One method of giving these instructions is to give commands through the keywords. The operating system working on commands are said to be command-driven.
  - OS can provide a GUI such as in Windows operating system, GNOME, KDE in which instructions are given through mouse clicks. This is a GUI-driven OS.

In embedded systems, the OS provides the same services as in desktop computers. However, as compared to desktops, embedded systems have special requirements as described in below :

- Reliability
- Multi-tasking with time constraints
- Small footprint
- Support diskless systems
- Portability
- Scalability
- Support for Standard API

- **Reliability:**
  - The OS in an embedded system has to be very reliable. The system should be available 99.999% of the time. The downtime per system for embedded is less than desktop computers. Downtime should occur rarely.
- **Multi-tasking with time constraints:**
  - If the embedded system need to conduct more than one work simultaneously, the system should be multitasking. Despite of this, the task management has to be done efficiently to meet the real time performance requirement. Desktop computers do not support real time requirements in most cases.

- **Small footprint:**
  - Since memory in the embedded system uses is limited, they have little memory for the OS itself. The memory occupied by the operating system is known as the ‘**footprint**’.
- **Support diskless systems:**
  - Embedded systems may not have secondary storage such as hard disk. The embedded OS along with application software will reside on a memory chip. File system management is not mandatory in embedded systems.

- **Portability:**
  - A variety of processors are available for developing embedded system.
  - There is no need to be reliable in the single brand for example of Intel in case of PC/Desktop. An important requirement of embedded operating systems is portability.
- **Scalability:**
  - The embedded operating systems may be used on an 8-bit micro-controller or a powerful 64-bit microprocessor. So, scalability is very important for embedded operating systems.
- **Support for standard API:**
  - Application software is developed using the Application Programming Interface (API) of the operating system.
  - An application developed for one OS may not be portable to another OS. To achieve portability, IEEE standardized the API called Portable Operating System Interface (POSIX). Operating systems used in embedded systems must comply with this standard.

Software Architecture

Activities of Embedded OS

- Task Scheduling
- Context Switching
- Mutual Exclusion
- Inter-task Communication
- Memory Management
- Timer Services

Software Architecture

Activities of Embedded OS

- Task Scheduling
  - Task have to share the CPU time in disciplined way, so that every task will get appropriate time.
  - Important time critical task have to be given high priority.
  - Provides mechanism to decide which task will get the next CPU time.
  - A number of task scheduling algorithm are available.
  - Deterministic Operating System :OS where time required to execute a task is estimated.

Software Architecture

Activities of Embedded OS

- Context Switching
  - If low priority task is being executed and high priority task is to be run.
  - In this case, CPU will be interrupted through interrupt by interrupt signal .
  - The CPU will save the current task information in stack and execute high priority task .

Software Architecture

Activities of Embedded OS

- Mutual Exclusion
  - While different task share the same resources, it is the responsibility of OS to ensure the events are mutually exclusive. For example: Two task have to share printer, the OS ensures that printer is shared without corrupting data.

Software Architecture

Activities of Embedded OS

- Inter-task Communication
  - Task may need to exchange the data . For example a task may write some data to a file and another task has to read that data .
  - Task have to synchronize their activities.
  - Special OS objects like, mailboxes, message queues, pipes, status register, and event flags are used to achieve inter task communication and inter task synchronization.

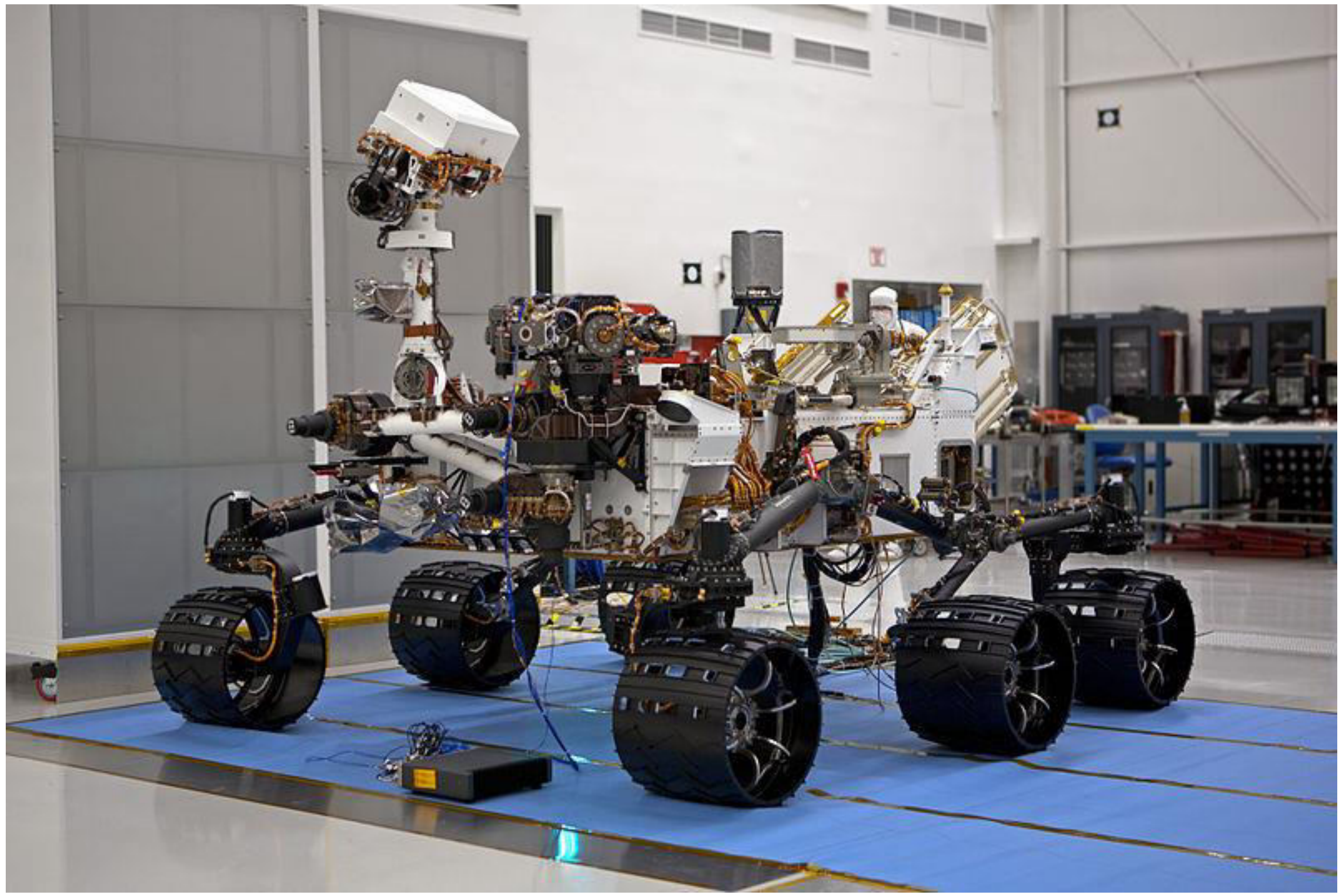
Software Architecture

Activities of Embedded OS

- Memory Management
  - Memory of Embedded system is to be shared by a number of tasks.
  - Memory Management is another service provided by OS.
- Timer Services
  - OS keep track of time for activities for which particular task is running.

Categories of Embedded Operating System

- Operating systems used in embedded systems can be broadly divided into the following categories:
  - Non-real time Embedded Operating System
    - Embedded Linux, Embedded NT, Windows XP Embedded
  - Real-time Operating Systems
    - QNX Neutrino, VxWorks, RTLinux, ….
  - Mobile/Handheld Operating System
    - Android, Palm OS, Symbian OS



- The Mars Science Laboratory Curiosity rover uses VxWorks

Application Software

- Application-specific software has to be developed above the operating system.
- There is need of development tools according to operating systems used.

- For example,
    - if we have to create a task to read data from a serial port, there is need to make function call to create new task with the required priority,
    - Write the code to read the data and write to a memory location. The various function calls provided by an operating system are:
        - To create, suspend and delete task
        - To do task scheduling for meeting real-time requirements
        - To facilitate inter-task communication and synchronization between tasks
        - To initialize, increment and reset counters to keep track of time
        - To allocate and free memory
        - To access the I/O devices
        - To access the communication protocol task.

- Using these function calls we can create the application software that is converted into a relocatable binary file and ported onto the program memory.
- At first executable file will be residiing in the EPROM or EEPROM or Flash memory.
- On power on, the program is transferred to the RAM and then the processor executes the program.
- Sometime the program may be executed directly from the program memory device. Such programs are called **Execute In Place (XIP)** programs.