# 1 Limited Direct Execution

**Vocabulary**

1. **Time Sharing**
   - Is a mechanism used by an OS to share a resource
   - Allows an entity to use the resource for a little while, and then a little while by another, and so forth

   **Example**

   CPU

2. **Limited Direct Execution**
   - Is synomyous to **baby proofing**
   - **Limited** Means there will be a limit to what a processor can and cannot do
   - **Direct Execution** Means that the processor will run directly on the CPU

3. **User Mode**
   - Is a processor mode where code that runs is restricted in what it can do

4. **Kernel Mode**
   - Is a processor mode where code that runs can do what it likes, including previleged operations

   **Example**

   Previleged operations include

   1. I/O requests
   2. Executing all types of restricted instructions

5. **System Call**
   - Is a programmatic way in which a computer program requests a previleged service from the kernel of the operating system

6. **Trap**
   - Is a type of synchronous interrupt caused by an exceptional condition that
   - Exceptional condition include:

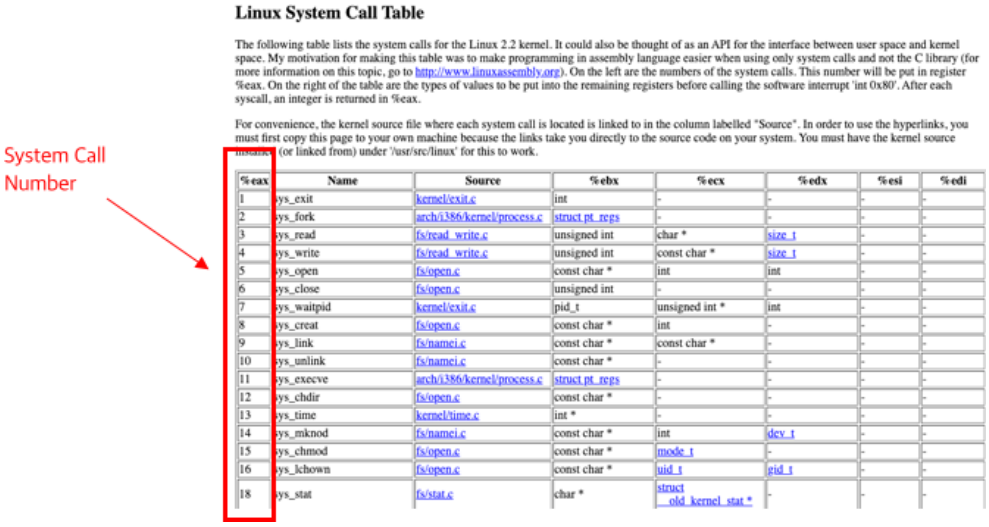
  - Breakpoint
  - Division by zero
  - Invalid memory access
  - System Call
- Usually results in a processor switching to kernel mode

7. **Return-from-Trap**
   - Is an instruction that
     - Restores saved registers from kernel stack
     - Swithces the processor back to **user mode**
8. **Trap Table**

Question What is the exact definition of a trap table? OSTEP glosses over it :(

- Is synonymous to 대응 메뉴얼
- Is a list of trap handlers where each is associated with a specific trap

9. **Trap Handlers**
   - Is the code that will run when the trap is triggered.
10. **System-call Number**
    - Is an ID assigned to each system call




11. **Timer Interrupt**
    - Is a type of interrupt generated by an internal clock instead of an external event (e.g I/O or system call)

12. **Interrupt Handler**
    - Is a special block of code associated with a specific interrupt condition

## 1.1 Direct Execution

-

## 1.2 Problem #1: Restricted Operations

- **Question:** How can the OS make sure a program doesn't do anything that we don't want to do while running it efficiently?

## 1.3 Problem #2: Switching Between Processes

- **Question:** When we are running a process, how does the oeprating system stop it from running and switch to another process, thus implementing **time sharing** mechanism to virtualize CPU?

## 1.4 Concurrency