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 for 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

Linux System Call Table

The following table lists the system calls for the Linux 2.2 kernel. It could also be thought of as an API for the interface between user space and kernel space. My motivation for making this table was to make programming in assembly language easier when using only system calls and not the C library (for more information on this topic, go to http://www.linuxassembly.org). On the left are the numbers of the system calls. This number will be put in register %eax. On the right of the table are the types of values to be put into the remaining registers before calling the software interrupt 'int 0x80'. After each syscall, an integer is returned in %eax.

For convenience, the kernel source file where each system call is located is linked to in the column labelled "Source". In order to use the hyperlinks, you must first copy this page to your own machine because the links take you directly to the source code on your system. You must have the kernel source installed (or linked from) under '/usr/src/linux' for this to work.

System Call Number →

%eax	Name	Source	%ebx	%ecx	%edx	%esi	%edi
1	ys_exit	kernel/exit.c	int	-	-	-	-
2	ys_fork	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
3	ys_read	fs/read_write.c	unsigned int	char *	size_t	-	-
4	ys_write	fs/read_write.c	unsigned int	const char *	size_t	-	-
5	ys_open	fs/open.c	const char *	int	int	-	-
6	ys_close	fs/open.c	unsigned int	-	-	-	-
7	ys_waitpid	kernel/exit.c	pid_t	unsigned int *	int	-	-
8	ys_creat	fs/open.c	const char *	int	-	-	-
9	ys_link	fs/namei.c	const char *	const char *	-	-	-
10	ys_unlink	fs/namei.c	const char *	-	-	-	-
11	ys_execve	arch/i386/kernel/process.c	struct pt_regs	-	-	-	-
12	ys_chdir	fs/open.c	const char *	-	-	-	-
13	ys_time	kernel/time.c	int *	-	-	-	-
14	ys_mknod	fs/namei.c	const char *	int	dev_t	-	-
15	ys_chmod	fs/open.c	const char *	mode_t	-	-	-
16	ys_lchown	fs/open.c	const char *	uid_t	gid_t	-	-
18	ys_stat	fs/stat.c	char *	struct old_kernel_stat *	-	-	-

11. **Timer Interrupt**

12. **Interrupt Handler**