Exceptions, Signals, Traps

- Special functions that the operating system implements
- Varies by Operating System

Exceptions and Faults

- Misbehaving Systems and Programs

Exception number	Description	Exception class
0	Divide error	Fault
13	General protection fault	Fault
14	Page fault	Fault
18	Machine check	Abort
32–255	OS-defined exceptions	Interrupt or trap

System Calls

- Functions that are provided by the operating system. Why are they so special?

Number	Name	Description	Number	Name	Description
0	`read`	Read file	33	`pause`	Suspend process until signal arrives
1	`write`	Write file	37	`alarm`	Schedule delivery of alarm signal
2	`open`	Open file	39	`getpid`	Get process ID
3	`close`	Close file	57	`fork`	Create process
4	`stat`	Get info about file	59	`execve`	Execute a program
9	`mmap`	Map memory page to file	60	`_exit`	Terminate process
12	`brk`	Reset the top of the heap	61	`wait4`	Wait for a process to terminate
32	dup2	Copy file descriptor	62	`kill`	Send signal to a process

System Calls: C and ASM

- Syscalls are a numbered list

```c
int main()
{
    write(1, "hello, world\n", 13);
    _exit(0);
}
```

code/ecf/hello-asm64.sa

```
.section .data
string:
  .ascii "hello, world\n"
string_end:
  .equ len, string_end - string
.section .text
.globl main
main:
  First, call write(1, "hello, world\n", 13)
  movq $1, %rax          write is system call 1
  movq $1, %rdi          Arg1: stdout has descriptor 1
  movq $string, %rsi     Arg2: hello world string
  movq $len, %rdx        Arg3: string length
  syscall                Make the system call

  Next, call _exit(0)
  movq $60, %rax         _exit is system call 60
  movq $0, %rdi          Arg1: exit status is 0
  syscall                Make the system call
```

code/ecf/hello-asm64.sa

List of Linux and FreeBSD Syscalls

- FreeBSD Syscall and Linux Syscall

```
/*
 * System call numbers.
 *
 * DO NOT EDIT-- this file is automatically @generated.
 */

#define         SYS_exit        SYS__exit

#define SYS_syscall     0
#define SYS__exit       1
#define SYS_fork        2
#define SYS_read        3
#define SYS_write       4
#define SYS_open        5
#define SYS_close       6
#define SYS_wait4       7
                                /* 8 is old creat */
#define SYS_link        9
#define SYS_unlink      10
                                /* 11 is obsolete execv */
#define SYS_chdir       12
#define SYS_fchdir      13
#define SYS_freebsd11_mknod     14
#define SYS_chmod       15
#define SYS_chown       16
#define SYS_break       17
                                /* 18 is freebsd4 getfsstat */
                                /* 19 is old lseek */
```

blob: 02bd6ddb627821aef22b2065066858b4b37bb177 (plain)

```
/* SPDX-License-Identifier: GPL-2.0-only */
/*
 * syscalls.h - Linux syscall interfaces (non-arch-specific)
 *
 * Copyright (c) 2004 Randy Dunlap
 * Copyright (c) 2004 Open Source Development Labs
 */

#ifndef _LINUX_SYSCALLS_H
#define _LINUX_SYSCALLS_H

struct __aio_sigset;
struct epoll_event;
struct iattr;
struct inode;
struct iocb;
struct io_event;
struct iovec;
struct __kernel_old_itimerval;
struct kexec_segment;
struct linux_dirent;
struct linux_dirent64;
struct list_head;
struct mmap_arg_struct;
struct msgbuf;
struct user_msghdr;
struct mmsghdr;
struct msqid_ds;
struct new_utsname;
struct nfsctl_arg;
struct __old_kernel_stat;
struct oldold_utsname;
struct old_utsname;
struct pollfd;
struct rlimit;
struct rlimit64;
```

Premptions and Concurrency

- The stack is back
- Old process is out
- New process comes in
- Excution resumes
- “Context Switch”