System Calls and Signals: Communication with the OS

Jonathan Misurda
jmisurda@cs.pitt.edu

Linux Syscalls

- 325 syscall slots reserved (2.6.23.1 kernel)
  - Not all are used

Syscall	Purpose
exit	Causes a process to terminate
fork	Creates a new process, identical to the current one
read	Reads data from a file or device
write	Writes data to a file or device
open	Opens a file
close	Closes a file
creat	Creates a file

Using Syscalls

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main()
{
        int fd;
        char buffer[100];
        strcpy(buffer, "Hello, World!\n");

        fd = open("hello.txt", O_WRONLY | O_CREAT);
        write(fd, buffer, strlen(buffer));
        close(fd);
        exit(0);

        return 0;
}
```

OR-ing Flags

- Define constants as powers of 2
- Bitwise OR to combine
- Bitwise AND to test

```
#define O_RDONLY     0
#define O_WRONLY     1
#define O_RDWR       2
#define O_CREAT     16
```

File Descriptors

- Integer identifying a unique open file
  - Similar to FILE *
- OS maintains additional information about the file to do things such as clean up on process termination
- Three standard file descriptors opened automatically:

  0 – stdin
  1 – stdout
  2 – stderr

fork()

- Creates a new process identical to the calling one
- Return value differs
  - “Child” process return value is 0
  - “Parent” process gets child’s process id number
- Often used with execv family of functions to launch a new program

Fork Example

```
#include <stdio.h>
#include <unistd.h>

int main()
{
        if(fork()==0)
        {
                printf("Hi from the child!\n");
        }
        else
        {
                printf("Hi from the parent\n");
        }

        printf("Hi from both\n");
        return 0;
}
```

Output

```
Hi from the child!
Hi from both
Hi from the parent
Hi from both
```

Spawning A Program

```
#include <stdio.h>
#include <unistd.h>

int main()
{
        if(fork()==0)
        {
                char *args[3] = {"ls“, "-al“, NULL};
                execvp(args[0], args);
        }
        else
        {
                int status;
                wait(&status);
                printf("Hi from the parent\n");
        }
        return 0;
}
```

Signals

- Notifications sent to a program by OS
  - Indicate special events
- Allows for asynchronous notification rather than polling
- Polling – to explicitly ask if something occurred, usually repeatedly

kill -l

SIGHUP	**SIGINT**	**SIGQUIT**	**SIGILL**	**SIGTRAP**
SIGABRT	**SIGBUS**	**SIGFPE**	**SIGKILL**	**SIGUSR1**
SIGSEGV	**SIGUSR2**	**SIGPIPE**	**SIGALRM**	**SIGTERM**
SIGCHLD	**SIGCONT**	**SIGSTOP**	**SIGTSTP**	**SIGTTIN**
SIGTTOU	**SIGURG**	**SIGXCPU**	**SIGXFSZ**	**SIGVTALRM**
SIGPROF	**SIGWINCH**	**SIGIO**	**SIGPWR**	**SIGSYS**
SIGRTMIN	SIGRTMIN+1	SIGRTMIN+2	SIGRTMIN+3	SIGRTMIN+4
SIGRTMIN+5	SIGRTMIN+6	SIGRTMIN+7	SIGRTMIN+8	SIGRTMIN+9
SIGRTMIN+10	SIGRTMIN+11	SIGRTMIN+12	SIGRTMIN+13	SIGRTMIN+14
SIGRTMIN+15	SIGRTMAX-14	SIGRTMAX-13	SIGRTMAX-12	SIGRTMAX-11
SIGRTMAX-10	SIGRTMAX-9	SIGRTMAX-8	SIGRTMAX-7	SIGRTMAX-6
SIGRTMAX-5	SIGRTMAX-4	SIGRTMAX-3	SIGRTMAX-2	SIGRTMAX-1
SIGRTMAX				