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

<table>
<thead>
<tr>
<th>Call</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td>Causes a process to terminate</td>
</tr>
<tr>
<td>fork</td>
<td>Creates a new process, identical to the calling one</td>
</tr>
<tr>
<td>read</td>
<td>Reads data from a file or device</td>
</tr>
<tr>
<td>write</td>
<td>Writes data to a file or device</td>
</tr>
<tr>
<td>open</td>
<td>Opens a file</td>
</tr>
<tr>
<td>close</td>
<td>Closes a file</td>
</tr>
<tr>
<td>creat</td>
<td>Creates a file</td>
</tr>
</tbody>
</table>

Using Syscalls

```c
#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!
    ");
    fd = open("hello.txt", O_WRONLY | O_CREAT);
    write(fd, buffer, strlen(buffer));
    close(fd);
    return 0;
}
```

OR-ing Flags

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

```c
#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 exec family of functions to launch a new program
Fork Example

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

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

Output

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

Spawning A Program

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

int main()
{
    if (fork()==0)
    {
        execvp(args[0], args);
    }
    else
    {
        int status;
        wait(&status);
        printf("Hi from the parent 
");
    }
    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