EECS 338-Introduction to Operating Systems

Recitation Session-2

January 25, 2011

Slides by E. Zeynep Erson

Recitation by Gary Doran

Today

- Java vs C
- C Basics
- Review system calls
  - fork, getpid,…
- More system calls
  - sleep, wait, exec, gethost..

Java vs C

You can use either, but I recommend C:

- I will assume you are using C in recitations
- C forces you to think like a machine (good practice for this course)
- The Kernel is written mostly in C
- Sometimes Java abstracts away too much

C Basics

- No boolean type (be careful)
- Strings can be tricky
- Memory Allocation Concerns

Strings in C

- String in C = array of characters terminated by \0 character.

For ex:

```
char msg[6];
msg[0]='F';
msg[1]='i'
…
```

Or

```
char msg[6] = {'F','i','r','s','t','\0'};
msg[0]='F';
msg[1]='i'
…
```

Or

```
char msg[20] = "First message" ;
```

String library functions

- char *strcpy(s,ct) -> copy ct into s, including ``\0''; return s
- char *strncpy(s,ct,n) -> copy ncharcater of ct into s, return s
- char *strcat(s,ct) -> concatenate ct to end of s; return s
- char *strncat(s,ct,n) -> concatenate n character of ct to end of s, terminate with ``\0''; return s
- int strcmp(cs,ct) -> compare cs and ct; return 0 if cs=ct, <0 if cs0 if cs>ct
- char *strchr(cs,c) -> return pointer to first occurence of c in cs or NULL if not encountered
- size_t strlen(cs) -> return length of cs

System calls, getting process ID

- Every process has its own, unique pid.
- Getting a process id:
  - Include: <sys/types.h>
    <unistd.h>
  - Signature: pid_t getpid()
  - Return:
    - Success: process ID
    - Failure -1

System calls, getting parent process ID

- Every process has associated parent process ID:
  - Include: <sys/types.h>
    <unistd.h>
  - Signature: pid_t getppid();
  - Returns:
    - Success: parent process ID
    - Failure: -1

System calls, gethostname( )

- #include <unistd.h>

int gethostname(char **name*, size_t *namelen*);

- Hostnames are limited to HOST_NAME_MAX

System Calls

- Creating a process by “fork” system call:
  - Include: <sys/types.h>
    <unistd.h>
  - Signature: pid_t fork()
  - Return:
    - Succes:0 in child, child pid in parent
    - Failure: -1

System calls, wait()

- wait for a child process to stop or terminate
- #include <sys/wait.h>
  pid_t wait(int *stat_loc);
- Stores information on *stat_loc*
- Returns pid of terminated child

System calls, exec(), execv(), execle(),..

- replaces the current process image with a new process image
- #include <unistd.h>
  - int execlp(const char *path, const char *arg0, const char *arg1, ... const char *argn, (char *) NULL);
- Ex. (note, by convention arg0 is the name of the program):

```
#include <unistd.h>
main() {
    execlp("ls", "ls", "-r", "-t", "-l", (char *) NULL);
}
```

System calls, sleep()

- #include <unistd.h>
  unsigned sleep(unsigned *seconds*);
  - Returns when *seconds* pass or signal is received
- int usleep(useconds_t *useconds*);
  - Returns when *useconds (microseconds)* pass or signal is received

time(), getenv(), putenv()

- See example code online: https://github.com/cwru-eecs338/syscall_examples