Recitation 9: Tshlab + VM

Instructor: TAs

Outline

- **Labs**
- **Signals**
- **IO**
- **Virtual Memory**

TshLab and MallocLab

- **TshLab due Tuesday**

- **MallocLab is released immediately after**
  - Start early
  - Do the checkpoint first, don't immediately go for the final
  - Expect a recitation next week
    - Working for several hours will improve the value significantly

Signals

- **Parent process sends SIGINT to a child process. What is the behavior of the child?**

- **What is the default?**
- **What else could the child do?**

More Signals

- **Parent process sends SIGKILL to a child process. What is the behavior of the child?**

- **What is the default?**
- **What else could the child do?**

Sending Signals

- **Parent sends SIGKILL to a child process.**

```
...
    pid_t pid = ...; // child pid
    kill(pid, SIGKILL);
    // At this point, what has happened
    //    to the child process?
```

Signals

- **How many times is Hi printed?**

```
int main(int argc, char** argv)
{
    pid_t ppid = getpid(), cpid, tpid;
    cpid = fork();
    if (cpid == 0) tpid = ppid;
    else tpid = cpid;
    kill(tpid, SIGINT);
    write(STDOUT_FILENO, "Hi", strlen("Hi"));
    return 0;
}
```

Blocking Signals

- **The shell is currently running its handler for SIGCHLD.**

- **What signals can it receive?**
- **What signals can it not receive (i.e., blocked)?**

Errno

- **Included from <errno.h>**

- **Global integer variable – usually 0**

- **When a system call fails (usually indicated by returning -1), it also will set errno to a value describing what went wrong**

- **Example: let's assume there is no "foo.txt" in our path**

```
int fd = open("foo.txt", O_RDONLY);
if(fd < 0) printf("%d\n", errno);
```

- **The code above will print 2 – in the man pages, we can see that 2 is ENOENT "No such file or directory"**

- **In shell lab, your signal handlers must preserve errno**

IO functions

Needed for tshlab

- `int open(const char *pathname, int flags);`
  - Some important flags:
  - O_CREAT – creates file if needed, opens for read/write
  - O_RDWR – opens for read/write
  - O_RDONLY – opens for read only
- `int close(int fd);`
- `int dup2(int oldfd, int newfd);`

Needed for life

- `ssize_t read(int fd, void *buf, size_t count);`
- `ssize_t write(int fd, const void *buf, size_t count);`
- `off_t lseek(int fd, off_t offset, int whence);`

IO and Fork()

- **File descriptor management can be tricky.**

- **How many file descriptors are open in the parent process at the indicated point?**

- **How many does each child have open at the call to execve?**

```c
int main(int argc, char** argv)
{
    int i;
    for (i = 0; i < 4; i++)
    {
        int fd = open("foo", O_RDONLY);
        pid_t pid = fork();
        if (pid == 0)
        {
            int ofd = open("bar", O_RDONLY);
            execve(...);
        }
    }
    // How many file descriptors are open in the parent?
```

Redirecting IO

■ **File descriptors can be directed to identify different open files.**

```
int main(int argc, char** argv)
{
    int i;
    for (i = 0; i < 4; i++)
    {
        int fd = open("foo", O_RDONLY);
        pid_t pid = fork();
        if (pid == 0)
        {
            int ofd = open("bar", O_WRONLY);
            dup2(fd, STDIN_FILENO);
            dup2(ofd, STDOUT_FILENO);
            execve(...);
        }
    }
    // How many file descriptors are open in the parent?
```

Redirecting IO

- **At the two points (A and B) in main, how many file descriptors are open?**

```
int main(int argc, char** argv)
{
    int i, fd;
    fd = open("foo", O_WRONLY);
    dup2(fd, STDOUT_FILENO);
    // Point A
    close(fd);
    // Point B
    ...
```

Memory Access

- **The processor tries to write to a memory address.**

- **List different steps that are required to complete this operation.**

Memory Access

- **The processor tries to write to a memory address.**

- **List different steps that are required to complete this operation. (non exhaustive list)**

- **Virtual to physical address conversion (TLB lookup)**

- **TLB miss**

- **Page fault, page loaded from disk**

- **TLB updated, check permissions**

- **L1 Cache miss (and L2 … and)**

- **Request sent to memory**

- **Memory sends data to processor**

- **Cache updated**

Address Translation with TLB

- **Translate 0x15213, given the contents of the TLB and the first 32 entries of the page table below.**

- **1MB Virtual Memory**
  256KB Physical Memory
  4KB page size

VPN	PPN	Valid	VPN	PPN	Valid
00	17	1	10	26	0
01	28	1	11	17	0
02	14	1	12	0E	1
03	0B	0	13	10	1
04	26	0	14	13	1
05	13	0	15	18	1
06	0F	1	16	31	1
07	10	1	17	12	0
08	1C	0	18	23	1
09	25	1	19	04	0
0A	31	0	1A	0C	1
0B	16	1	1B	2B	0
0C	01	0	1C	1E	0
0D	15	0	1D	3E	1
0E	0C	0	1E	27	1
0F	2B	1	1F	15	1

2-way set associative

Index	Tag	PPN	Valid
0	05	13	1
	3F	15	1
1	10	0F	1
	0F	1E	0
2	1F	01	1
	11	1F	0
3	03	2B	1
	1D	23	0

Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition

If you get stuck on TshLab

- **Read the writeup!**
- **Do manual unit testing before `runtrace` and `sdriver`!**
- **Post private questions on piazza!**

- **Read the man pages on the syscalls.**
  - Especially the error conditions
  - What errors should terminate the shell?
  - What errors should be reported?