CS 134: Operating Systems
More Synchronization
Overview

Homework Discussion
Questions About Unix

We should have talked about this last Thursday:

1. What does `getpid` do?

2. What does `stime` do?

3. What is interesting about `readdir`?

4. How many system calls does Linux have?

5. What did you run `strace` on, and what did you learn?
We should have talked about this last Thursday:

1. What does `getpid` do?  
   Returns the current process ID

2. What does `stime` do?  
   Sets the time of day. What might that involve?

3. What is interesting about `readdir`?  
   "This is not the function you are interested in." . . . nor is `getdents`.

4. How many system calls does Linux have?  
   About 440.

5. What did you run `strace` on, and what did you learn?
Questions About Unix

We should have talked about this last Thursday:

1. **What does getpid do?**
   Returns the current process ID

2. **What does stime do?**
   Sets the time of day. What might that involve?

3. **What is interesting about readdir?**
   "This is not the function you are interested in." . . . nor is getdents.

4. **How many system calls does Linux have?**
   About 440.

5. **What did you run strace on, and what did you learn?**
Questions About Unix

We should have talked about this last Thursday:

1. **What does getpid do?**
   Returns the current process ID

2. **What does stime do?**
   Sets the time of day. What might that involve?

3. **What is interesting about readdir?**
   “This is not the function you are interested in.”

4. **How many system calls does Linux have?**

5. **What did you run strace on, and what did you learn?**
Questions About Unix

We should have talked about this last Thursday:

1. **What does getpid do?**
   Returns the current process ID

2. **What does stime do?**
   Sets the time of day. What might that involve?

3. **What is interesting about readdir?**
   “This is not the function you are interested in.”
   ... nor is getdents.

4. **How many system calls does Linux have?**
   About 440.

5. **What did you run strace on, and what did you learn?**
Questions About Unix

We should have talked about this last Thursday:

1. What does `getpid` do?
   Returns the current process ID

2. What does `stime` do?
   Sets the time of day. What might that involve?

3. What is interesting about `readdir`?
   "This is not the function you are interested in."
   ... nor is `getdents`.

4. How many system calls does Linux have?
   About 440.

5. What did you run `strace` on, and what did you learn?
1. The kernel for this assignment is configured to use a particular VM system. What is this VM system called?

2. Which register number is used for the stack pointer \((sp)\) in OS/161?

3. What bus/busses does OS/161 support?
1. The kernel for this assignment is configured to use a particular VM system. What is this VM system called? dumbvm from kern/arch/mips/conf/conf.arch

2. Which register number is used for the stack pointer (sp) in OS/161?

#define sp $29 /* stack pointer */ from kern/arch/mips/include/asmdefs.h

3. What bus/busses does OS/161 support?

LAMEbus from kern/arch/mips/include/bus.h
1. The kernel for this assignment is configured to use a particular VM system. What is this VM system called?
   dumbvm from kern/arch/mips/conf/conf.arch

2. Which register number is used for the stack pointer (sp) in OS/161?
   #define sp $29 /* stack pointer */
   from kern/arch/mips/include/asmdefs.h

3. What bus/busses does OS/161 support?
   LAMEbus from kern/arch/mips/include/bus.h
1. The kernel for this assignment is configured to use a particular VM system. What is this VM system called?
   dumbvm from kern/arch/mips/conf/conf.arch

2. Which register number is used for the stack pointer (\texttt{sp}) in OS/161?
   
   \begin{verbatim}
   #define sp $29 /* stack pointer */
   \end{verbatim}
   from kern/arch/mips/include/asmdefs.h

3. What bus/busses does OS/161 support?
   
   LAMEbus from kern/arch/mips/include/bus.h
4. What is the difference between `splhigh` and `spl0`?

5. Why do we use `typedef` like `u_int32_t` instead of simply saying `int`?

6. What must be the first thing in the process control block?
4. What is the difference between `splhigh` and `spl0`?
   `splhigh()` sets `spl` to the highest value, disabling all interrupts
   `spl0()` sets `spl` to 0, enabling all interrupts

5. Why do we use `typedef` like `u_int32_t` instead of simply saying `int`?
   To make sure that we really get a 32-bit unsigned integer (`unsigned int` depends on the platform)

6. What must be the first thing in the process control block?
   "Note that `pcb_switchstack` MUST BE THE FIRST THING IN THE PCB or `switch.S` will have a coronary."
4. What is the difference between `splhigh` and `spl0`?
   `splhigh()` sets spl to the highest value, disabling all interrupts
   `spl0()` sets spl to 0, enabling all interrupts
   
   from kern/arch/mips/include/spl.h

5. Why do we use typedefs like `u_int32_t` instead of simply saying `int`?
   To make sure that we really get a 32-bit unsigned integer
   (unsigned int depends on the platform)
   related to kern/arch/mips/include/types.h

6. What must be the first thing in the process control block?
   *Note that `pcb_switchstack` MUST BE THE FIRST THING IN THE PCB or `switch.S` will have a coronary.*
   from kern/arch/mips/include/pcb.h
4. What is the difference between `splhigh` and `spl0`?
   - `splhigh()` sets spl to the highest value, disabling all interrupts
   - `spl0()` sets spl to 0, enabling all interrupts

   from kern/arch/mips/include/spl.h

5. Why do we use typedefs like `u_int32_t` instead of simply saying `int`?
   To make sure that we really get a 32-bit unsigned integer
   (unsigned int depends on the platform)

   related to kern/arch/mips/include/types.h

6. What must be the first thing in the process control block?
   “Note that `pcb_switchstack` MUST BE THE FIRST THING IN THE PCB or `switch.S` will have a coronary.”

   from kern/arch/mips/include/pcb.h
7. What does `splx` return?

8. What is the highest interrupt level?

9. What function is called when user-level code generates a fatal fault?

10. How frequently are “hardclock” interrupts generated?
7. What does `splx` return? 
The old interrupt state  
*from* kern/arch/mips/mips/spl.c

8. What is the highest interrupt level? 
`#define SPL_HIGH 15`  
*from* kern/arch/mips/include/spl.h

9. What function is called when user-level code generates a fatal fault? 
`kill_curthread`  
*from* kern/arch/mips/mips/trap.c

10. How frequently are “hardclock” interrupts generated? 
`#define HZ 100`  
i.e., 100 times a second  
*from* kern/include/clock.h

2013-05-19
CS34 Homework Discussion
Written Answers (3)
7. What does `splx` return? The old interrupt state
   from kern/arch/mips/mips/spl.c

8. What is the highest interrupt level?
   `#define SPL_HIGH 15`
   from kern/arch/mips/include/spl.h

9. What function is called when user-level code generates a fatal fault?

10. How frequently are “hardclock” interrupts generated?
7. What does `splx` return? 
The old interrupt state 

`from kern/arch/mips/mips/spl.c`

8. What is the highest interrupt level? 

`#define SPL_HIGH 15`

`from kern/arch/mips/include/spl.h`

9. What function is called when user-level code generates a fatal fault? 

`kill_curthread` `from kern/arch/mips/mips/trap.c`

10. How frequently are “hardclock” interrupts generated? 

`#define HZ 100`

i.e., 100 times a second

`from kern/include/clock.h`
7. What does \texttt{splx} return?
The old interrupt state \\
\texttt{from} kern/arch/mips/mips/spl.c

8. What is the highest interrupt level?
\texttt{#define SPL\_HIGH 15} \\
\texttt{from} kern/arch/mips/include/spl.h

9. What function is called when user-level code generates a fatal fault?
\texttt{kill\_curthread} \texttt{from} kern/arch/mips/mips/trap.c

10. How frequently are “hardclock” interrupts generated?
\texttt{#define HZ 100} \\
i.e., 100 times a second \texttt{from} kern/include/clock.h
11. What functions comprise the standard interface to a VFS device?

12. How many characters are allowed in an SFS volume name?

13. What is the standard interface to a file system (i.e., what functions must you implement to implement a new file system)?
11. What functions comprise the standard interface to a VFS device?
   d_open, d_close, d_io, d_ioctl
   from kern/include/dev.h

12. How many characters are allowed in an SFS volume name?
   #define SFS_VOLNAME_SIZE 32 /* max length of volume name */
   from kern/include/kern/sfs.h

13. What is the standard interface to a file system (i.e., what functions must you implement to implement a new file system)?
11. What functions comprise the standard interface to a VFS device?
   d_open, d_close, d_io, d_ioctl
   from kern/include/dev.h

12. How many characters are allowed in an SFS volume name?
    #define SFS_VOLNAME_SIZE 32 /* max length of volume name */
    from kern/include/kern/sfs.h

13. What is the standard interface to a file system (i.e., what functions must you implement to implement a new file system)?
11. What functions comprise the standard interface to a VFS device?
   d_open, d_close, d_io, d_ioctl
   from kern/include/dev.h

12. How many characters are allowed in an SFS volume name?
   #define SFS_VOLNAME_SIZE 32 /* max length of volume name */
   from kern/include/kern/sfs.h

13. What is the standard interface to a file system (i.e., what functions must you implement to implement a new file system)?
   fs_sync, fs_getvolname, fs_getroot, fs_umount
   from kern/include/fs.h
14. What function puts a thread to sleep?

15. How large are OS/161 pids?

16. What operations can you perform on a vnode?
14. What function puts a thread to sleep?
   `void thread_sleep(const void *addr);
   from kern/include/thread.h`

15. How large are OS/161 pids?
   `typedef int32_t pid_t; /* Process ID */
   32 bits / 4 bytes
   from kern/include/kern/types.h`

16. What operations can you perform on a `vnode`?
   `open, close, reclaim, read, readlink, getdirentry, write, ioctl, stat, gettype, tryseek, fsync, mmap, truncate, namefile, creat, symlink, mkdir, link, remove, rmdir, rename, lookup, lookparent`
   from kern/include/vnode.h
14. What function puts a thread to sleep?

```c
void thread_sleep(const void *addr);
from kern/include/thread.h
```

15. How large are OS/161 pids?

```c
typedef int32_t pid_t; /* Process ID */
32 bits / 4 bytes from kern/include/kern/types.h
```

16. What operations can you perform on a vnode?

```c
open, close, reclaim, read, readlink, getdirentry, write, ioctl, stat, gettype, tryseek, fsync, mmap, truncate, namefile, creat, symlink, mkdir, link, remove, rmdir, rename, lookup, lookparent
from kern/include/vnode.h
```
14. What function puts a thread to sleep?

```c
void thread_sleep(const void *addr);
```

from kern/include/thread.h

15. How large are OS/161 pids?

```c
typedef int32_t pid_t; /* Process ID */
```

32 bits / 4 bytes from kern/include/kern/types.h

16. What operations can you perform on a vnode?

- open, close, reclaim, read, readdir, getdirentry, write, ioctl, stat, gettype, tryseek, fsync, mmap, truncate, namefile, creat, symlink, mkdir, link, remove, rmdir, rename, lookup, lookparent

from kern/include/vnode.h
17. What is the maximum path length in OS/161?

18. What is the system call number for a reboot?

19. Where is `STDIN_FILENO` defined?
17. What is the maximum path length in OS/161?

/* Longest full path name */
#define PATH_MAX 1024

from kern/include/kern/limits.h

18. What is the system call number for a reboot?

#define SYS_reboot 8 /* Reboot system */
from kern/include/kern/callno.h

(c.f., RB_REBOOT in kern/include/kern/unistd.h)

19. Where is STDIN_FILENO defined?

#define STDIN_FILENO 0 /* Standard input */
from kern/include/kern/unistd.h
17. What is the maximum path length in OS/161?
   /* Longest full path name */
   #define PATH_MAX 1024
   from kern/include/kern/limits.h

18. What is the system call number for a reboot?
   #define SYS_reboot 8 /* Reboot system */
   from kern/include/kern/callno.h
   (c.f., RB_REBOOT in kern/include/kern/unistd.h)

19. Where is STDIN_FILENO defined?
   #define STDIN_FILENO 0 /* Standard input */
   from kern/include/kern/unistd.h
17. What is the maximum path length in OS/161?
   /* Longest full path name */
   #define PATH_MAX 1024
   from kern/include/kern/limits.h

18. What is the system call number for a reboot?
   #define SYS_reboot 8 /* Reboot system */
   from kern/include/kern/callno.h
   (c.f., RB_REBOOT in kern/include/kern/unistd.h)

19. Where is STDIN_FILENO defined?
   #define STDIN_FILENO 0 /* Standard input */
   from kern/include/kern/unistd.h
20. **What does kmain() do?**

21. **Is it okay to initialize the thread system before the scheduler?**
   Why or why not?

22. **What is a zombie?**

23. **How large is the initial run queue?**
20. What does `kmain()` do?
   Kernel main. (Boot up, then fork the menu thread, wait for a reboot request, and then shut down.)
   
   *from* kern/main/main.c

21. Is it okay to initialize the thread system before the scheduler? Why or why not?

22. What is a zombie?

23. How large is the initial run queue?
20. **What does kmain() do?**
   Kernel main. (Boot up, then fork the menu thread, wait for a reboot request, and then shut down.)

   *from* kern/main/main.c

21. **Is it okay to initialize the thread system before the scheduler? Why or why not?**
   Yes. The scheduler bootstrap just creates the run queue, and the thread bootstrap just initializes the first thread.

22. **What is a zombie?**

23. **How large is the initial run queue?**
20. What does \texttt{kmain()} do?
   Kernel main. (Boot up, then fork the menu thread, wait for a
   reboot request, and then shut down.)
   \textit{from} kern/main/main.c

21. Is it okay to initialize the thread system before the scheduler?
   Why or why not?
   Yes. The scheduler bootstrap just creates the run queue, and
   the thread bootstrap just initializes the first thread.

22. What is a zombie?
   “Zombies are threads/processes that have exited but not
   been fully deleted yet.” \textit{from} kern/thread/thread.c

23. How large is the initial run queue?
20. What does `kmain()` do?

Kernel main. (Boot up, then fork the menu thread, wait for a reboot request, and then shut down.)

`from kern/main/main.c`

21. Is it okay to initialize the thread system before the scheduler? Why or why not?

Yes. The scheduler bootstrap just creates the run queue, and the thread bootstrap just initializes the first thread.

22. What is a zombie?

"Zombies are threads/processes that have exited but not been fully deleted yet."  

`from kern/thread/thread.c`

23. How large is the initial run queue?

`runqueue = q_create(32);`

`from kern/thread/scheduler.c`
24. Can an array represented by a `struct array` be resized?

Yes, using `array_setsize` from `kern/lib/array.c` see also `kern/include/array.h`.

25. What does a device name in OS/161 look like?

The name of a device is always `device:`, such as `lhd0:` from `kern/fs/vfs/device.c`.

26. What does a raw device name in OS/161 look like?

The name with `raw` appended, such as `lhd0raw:` from `kern/fs/vfs/vfslist.c`.

27. What lock protects the vnode reference count?

`vn_countlock` from `kern/fs/vfs/vnode.c`.

28. What device types are currently supported?

Block & character devices. from `kern/fs/vfs/device.c`.
24. Can an array represented by a \texttt{struct array} be resized? Yes, using \texttt{array_setsize}. \textit{from} kern/lib/array.c \textit{see also} kern/include/array.h

25. What does a device name in OS/161 look like?

26. What does a raw device name in OS/161 look like?

27. What lock protects the vnode reference count?

28. What device types are currently supported?
24. Can an array represented by a struct array be resized?
   Yes, using array_setsize.  
   from kern/lib/array.c
   see also kern/include/array.h

25. What does a device name in OS/161 look like?
   The name of a device is always device:, such as lhd0:  
   from kern/fs/vfs/device.c

26. What does a raw device name in OS/161 look like?

27. What lock protects the vnode reference count?

28. What device types are currently supported?
24. Can an array represented by a struct array be resized?  
Yes, using array_setsize.  from kern/lib/array.c  
see also kern/include/array.h

25. What does a device name in OS/161 look like?  
The name of a device is always device:, such as lhd0:  
from kern/fs/vfs/device.c

26. What does a raw device name in OS/161 look like?  
The name with raw appended, such as lhd0raw:  
from kern/fs/vfs/vfslist.c

27. What lock protects the vnode reference count?  

28. What device types are currently supported?  

Block & character devices.  from kern/fs/vfs/device.c
24. Can an array represented by a `struct array` be resized? Yes, using `array_setsize`.  
   *from* kern/lib/array.c  
   *see also* kern/include/array.h

25. What does a device name in OS/161 look like?  
The name of a device is always `device:` such as `lhd0:`  
   *from* kern/fs/vfs/device.c

26. What does a raw device name in OS/161 look like?  
The name with `raw` appended, such as `lhd0raw:`  
   *from* kern/fs/vfs/vfslist.c

27. What lock protects the vnode reference count?  
   `vn_countlock`  
   *from* kern/fs/vfs/vnode.c

28. What device types are currently supported?  
   Block & character devices.  
   *from* kern/fs/vfs/device.c
24. Can an array represented by a struct array be resized? Yes, using array_setsize. from kern/lib/array.c

see also kern/include/array.h

25. What does a device name in OS/161 look like? The name of a device is always device:, such as lhd0: from kern/fs/vfs/device.c

26. What does a raw device name in OS/161 look like? The name with raw appended, such as lhd0raw: from kern/fs/vfs/vfslist.c

27. What lock protects the vnode reference count? vn_countlock from kern/fs/vfs/vnode.c

28. What device types are currently supported? Block & character devices. from kern/fs/vfs/device.c