

Today's Class

- **Last time (Tues Nov 20)**
 - Assignment #5 due
 - Assignment #6 out (due Thurs Nov 29)
 - File systems implementation (Chapts 11/12)
- **This time (Tues Nov 27)**
 - PA#6 discussion
 - Maybe I/O systems, Mass-storage systems
 - PA #7 out (due last day of class – Thurs Dec 6)
- **Next time (Thurs Nov 29)**
 - Protection/security
 - Assignment #6 due
 - PA#7 discussion
- **Extra class this week (Thurs 6pm/Fri 3pm)**
- **Course Evaluations now available!**

CS414: Operating Systems

Before we start

- **Blake's office hours (002a) for the rest of the semester:**
 - Wed this week 5-8
 - Mon next week 5-8
 - Wed next week 5-8

CS414: Operating Systems

Schedule

Sun	Tues	Thurs	Fri
	Mass storage 27	protection, security PA#6 due, PA#7 out protection, security 29	protection, security 30
	protection, security 4	Wrap-up PA#7 due 6	
		Final exam 10:00-noon 14	

CS414: Operating Systems

PA#6

- **On 002a machines: use cygwin, gcc -Wall hw6.c (c is probably advised)**
 - Seg fault? "gcc -g -Wall hw6.c, gdb ./a.exe, run"
- **How many inodes in 1 block?**
 - floor (BLOCK_SIZE / sizeof (struct inode))
- **fseek**
- **Say you have a char block[1024], how to determine if it's an inode block?**
 - struct inode * Potential_inode = (struct inode *) &block[0];
 - Potential_inode->uid
 - Note: you have to look through all potential inode memory locations (if you don't find one in position 1, you could still find one in position 2 – if, for example, the file in position 1 was deleted)

CS414: Operating Systems

PA#6

- **Advice:**
 - [1] write a routine to print an inode (make sure you understand the structure – particularly the iblocks)
 - assert(potential_block_num > 0);
 - assert (potential_block_num < MAX_BLOCK_NUM);
 - [2] then look for an inode in the datafile (and print it out)
 - [3] write out the file pointed at by the inode
 - [4] on the command prompt, invoke "file" (embedding this programmatically – e.g., via FORK – is better)

CS414: Operating Systems

PA #6

	Xeon, linux	Athlon, linux	AMD processor 242, linux x86_64	Laptop (Intel T2050), win, gcc	Laptop, win, VS
Char	1	1	1	1	1
Char *	4	4	8	4	4
Int	4	4	4	4	4
Short	2	2	2	2	2
Long	4				
Float	4	4	4	4	4
Double	8	8	8	8	8

CS414: Operating Systems

PA#6

•Endianness

- Assume we have the value 0x0A0B0C0D
- Little-Endian (increasing numeric significance with increasing memory addresses)
 - Address a: 0D, Address a+1: 0C
- Big-Endian ("big end first": most-significant byte first)
 - Address a: 0A, Address a+1: 0B

•Test:

```
int am_big_endian()
{
    long one= 1;
    return !*((char *)&one);
}
```