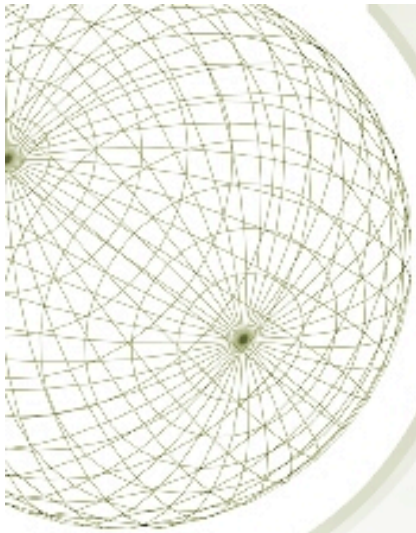
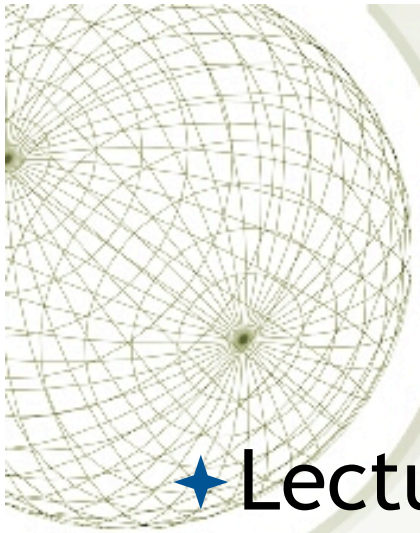


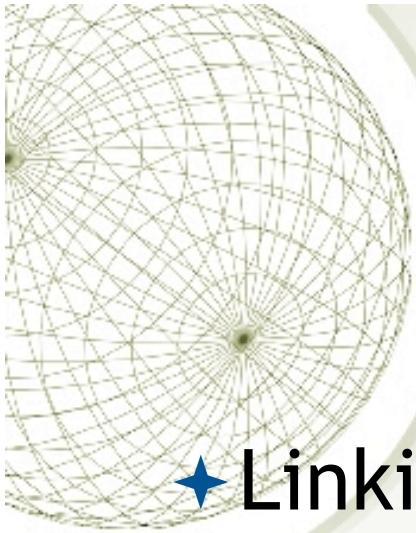
213 Recitation
Exam 2 review

Elie Krevat and Jiri Simsa



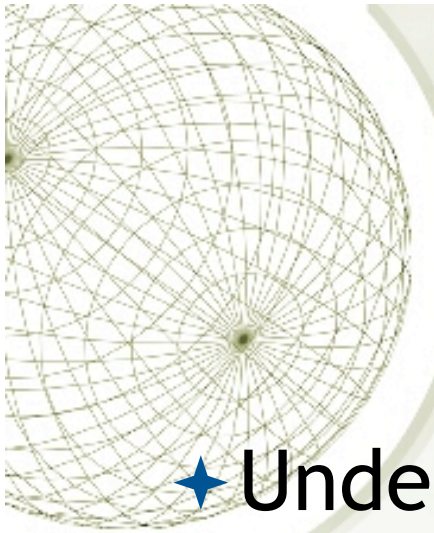
Exam coverage

- Lectures 8-17, Buf lab, Shell lab:
  - Buffer overflow
  - Main Memory and Caches
  - Exceptions & Logical control flow
  - Sys-level I/O
  - Virtual Memory
  - Malloc
  - Disk Storage



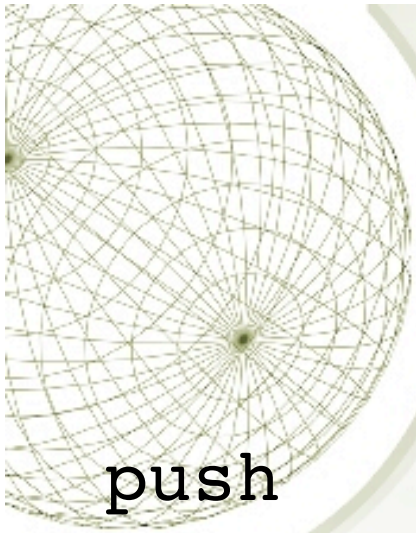
What's NOT covered

- Linking
- Garbage Collection (not in depth)
- VM access permissions (not in depth)
- 2-level Page Tables



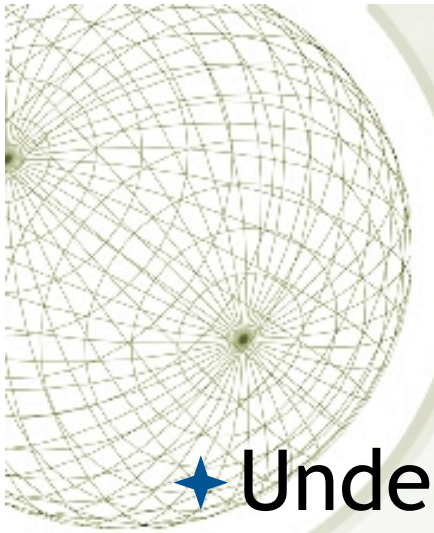
Buffer overflow

- Understand layout of a stack frame
- Be able to read assembly code
  - Frame ptr, stack ptr, instruction ptr
- Given: unsafe code in C and assembly
- Output: exploit string that calls a function



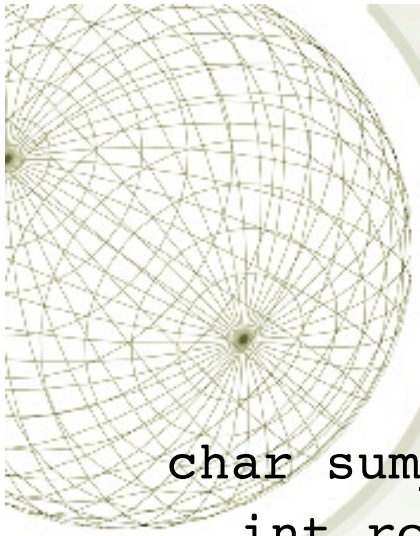
Buffer overflow

```
push   %rbp
mov    %rsp,%rbp
sub    $0x10,%rsp
lea    0xfffffffffffffff0(%rbp),%rdi
mov    $0x0,%eax
  callq  4003a8 <gets@plt>
mov    $0x0,%eax
leaveq
retq
```



Caches

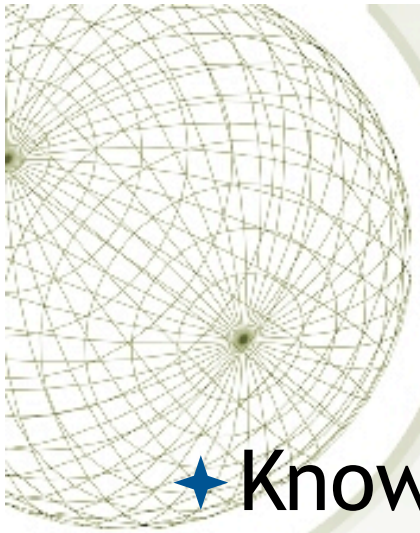
- Understand operation of caches
  - Bits for tag, set index, block offset
  - Direct-mapped, fully associative, or N-way set associative
  - Coding for locality & replacement policies
- Given: Cache description and sequence of memory accesses
- Output: Identify hits and misses



Caches

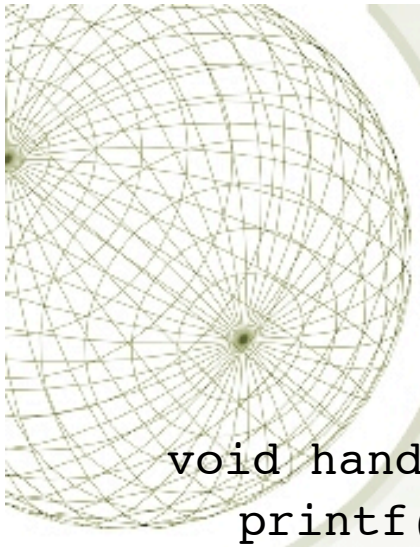
```
char sum_matrix(char matrix[3][3]) {
    int row, col;
    char sum = 0;
    for (col = 0; col < 3; col++) {
        for (row = 0; row < 3; row++) {
            sum += matrix[row][col];
        }
    }
    return sum;
}
```

- chars are 8 bits
- cache is direct-mapped with 4 sets of 4 bytes



Control flow & signals

- Know fork(), wait(), kill(), signal(), ...
- Given: C source code with printf()
- Output: All possible outputs

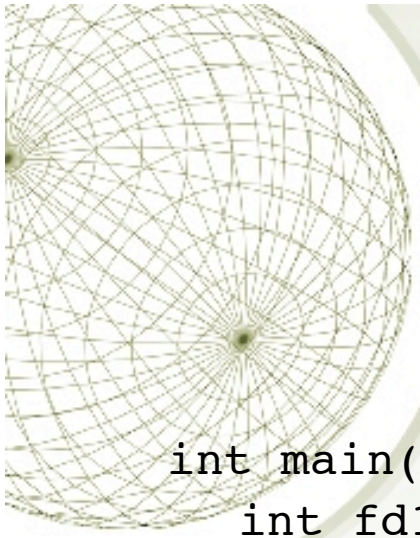


Control flow & signals

```
void handler(int sig) {
   printf("Whoops.\n");
   exit(0);
}

int main() {
   int pid;
   signal(SIGUSR1,handler);
   if (pid = fork()) {
       printf("Hi!\n");
       kill(pid,SIGUSR1);
   }
   printf("Peace\n");
}
```

Sys-level I/O

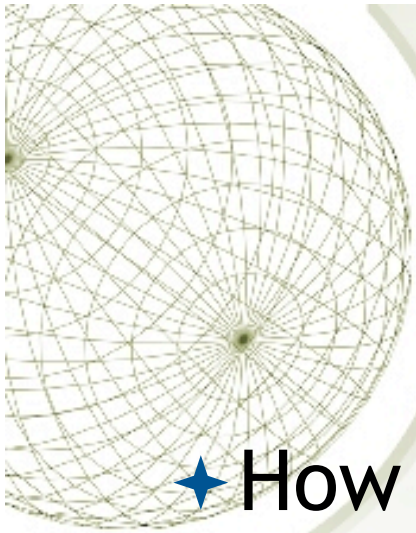
- File descriptors!
- I/O redirection
- Descriptor table per-process
- Open file/vnode table all processes
- Know open(), dup2(), fork(), ...
- Given: C sources with printf()
- Output: Correct output



File descriptors

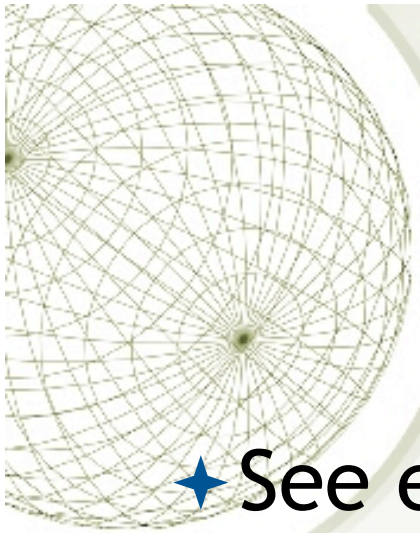
```
int main() {
    int fd1, fd2;
    char c;

    int fd1 = open("test.txt",RD_ONLY);
    fd2 = dup(fd1);
    read(fd2,&c,1);
    printf("1 = %c.\n",c);
    fork();
    read(fd1,&c,1);
    printf("2 = %c.\n",c);
}
```

- test.txt contains "15213 rocks!"



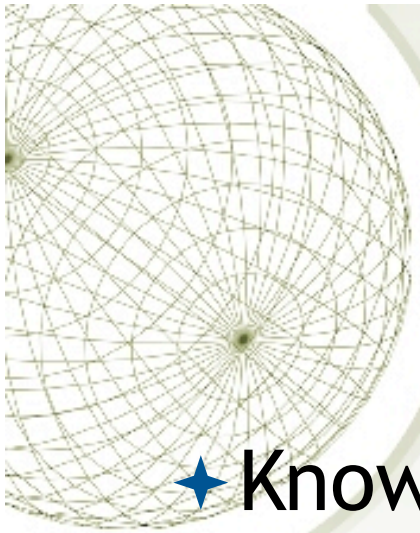
Virtual memory

- How memory request is handled
  - Translation look-aside buffer
  - Virtual vs. physical address
  - Page table
  - Page hit vs. Page miss
  - Given: VM description and requests
  - Output: Sequence of events



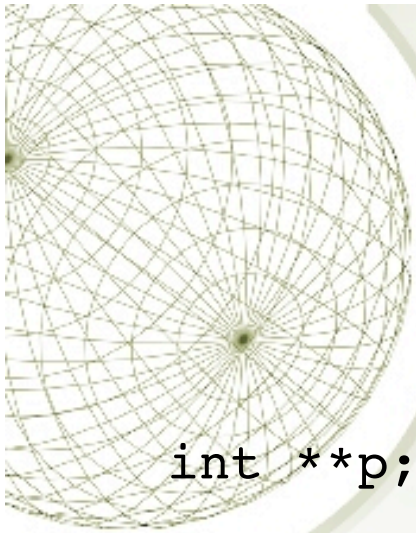
Virtual memory

- See exam 2, fall ‘06, problem 4



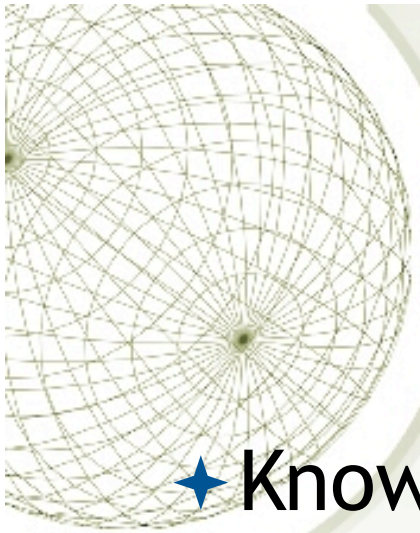
Malloc()

- Know malloc()
  - Implicit/explicit list (free blocks)
  - Types of errors, e.g.:
    - mem leaks
    - wrong allocation amounts
    - Initialization
  - Input: C code
  - Output: Identify error



Malloc()

```
int **p;

p = malloc(N*sizeof(int));

for (i=0; i<N; i++) {
   p[i] = malloc(M*sizeof(int));
}
```



Disk Storage

- Know physical construction of disk
  - Disk heads, platters, etc.
- How data is read
- Speed and performance characteristics

Questions?

- Good luck!