Lab1 notes

- Due Friday
- Hint: Read the manual pages (esp. for pipe(2))
- Best if last process in pipeline is child of shell
  - Consider: `cat file | grep word`
  - grep should be a child of ash (shell)
  - Don’t want to read next command until grep is done
  - In background, care more about exit status of grep
  - OK to make cat a child of grep:
    ```
    if (!fork ()) {
        /* set up pipe */
        if (!fork ()) /* exec cat */;
        /* exec grep */;
    }
    ```
Example memory layout

4 Gig

First 256MB physical memory
kernel text & most data

mapped kernel data
Invalid Memory

user stack

[mmaped regions]

break point

heap
BSS
program data
program text (read-only)
Invalid Memory
Paging in day-to-day use

- Demand paging
- Growing the stack
- BSS page allocation
- Shared text
- Shared libraries
- Shared memory
- Copy-on-write (fork, mmap, etc.)
- Q: Which pages should have global bit set on x86?
mmap system call

- **void *mmap (void *addr, size_t len, int prot,**
  int flags, int fd, off_t offset)

  - prot: OR of PROT_EXEC, PROT_READ, PROT_WRITE, PROTNONE
  - flags: OR of MAP_FIXED, MAP_ANON, MAP_PRIVATE, MAP_SHARED, ...
  - Private means other processes don’t see changes until msync
  - Q: Can mmap always have exact desired semantics?

- **int msync(void *addr, size_t len, int flags)**
  - If len == 0, do entire mmapped region
  - flags: MSASYNC, MS_SYNC, MS_INVALIDATE
More system calls

- `int munmap(void *addr, size_t len)`
  - Removes memory-mapped object

- `int mprotect(void *addr, size_t len, int prot)`
  - Changes protection on pages to `prot` (OR of PROT_xxx)
  - Q: What must OS do to implement on x86 hardware?

- `int mincore(void *addr, size_t len, char *vec)`
  - Returns in `vec` which pages present

- `int mlock (void *addr, size_t len)`
- `int munlock (void *addr, size_t len)`
  - Pin/unpin pages in memory
Catching page faults

struct sigaction {
    union { /* signal handler */
        void (*sa_handler)(int);
        void (*sa_sigaction)(int, siginfo_t *, void *);
    };
    sigset_t sa_mask; /* signal mask to apply */
    int sa_flags;
};

int sigaction (int sig, const struct sigaction *act,
               struct sigaction *oact)

• Can specify function to run on SIGSEGV
Example: OpenBSD/i386 siginfo

```c
struct sigcontext {
    int sc_gs; int sc_fs; int sc_es; int sc_ds;
    int sc edi; int sc esi; int sc ebp; int sc ebx;
    int sc edx; int sc ecx; int sc eax;

    int sc eip; int sc cs; /* instruction pointer */
    int sc eflags;         /* condition codes, etc. */
    int sc esp; int sc ss; /* stack pointer */

    int sc onstack;       /* sigstack state to restore */
    int sc mask;          /* signal mask to restore */

    int sc trapno;
    int sc err;
};
```