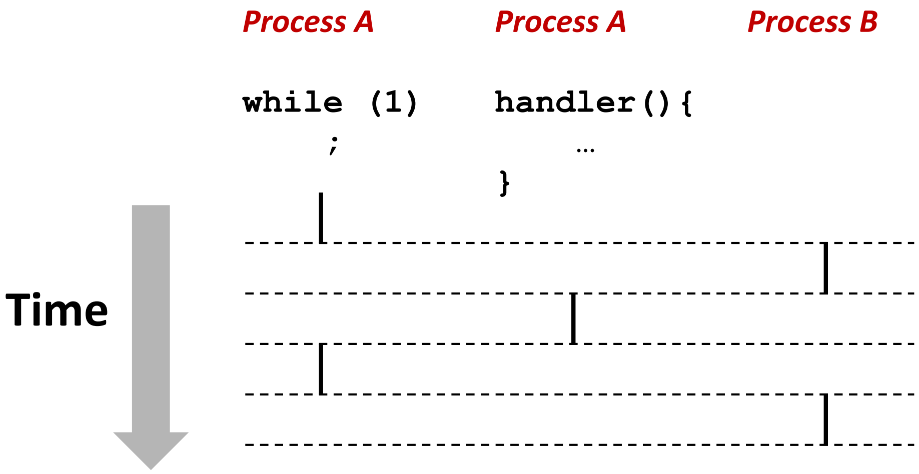
Signal Handler as Concurrent Flow



Concurrency Example (1)

```c
int main(int argc, char** argv) {
    int pid;

    Signal(SIGCHLD, handler);
    initjobs(); /* Initialize the job list */

    while (1) {
        if ((pid = fork()) == 0) { /* Child */
            execve("/bin/date", argv, NULL);
        }
        addjob(pid);  /* Add child to job list */
    }
    exit(0);
}
```

```c
void handler(int sig) {
    pid_t pid;

    while ((pid = waitpid(-1, NULL, 0)) > 0) { /* Reap child */
        deletejob(pid); /* Delete the child from the job list */
    }
    if (errno != ECHILD)
        unix_error("waitpid error");
}
```

Concurrency Example (2)

```c
int main(int argc, char** argv) {
    int pid;
    sigset_t mask_all, prev_all;
    sigfillset(&mask_all);
    Signal(SIGCHLD, handler);
    initjobs(); /* Initialize the job list */
    while (1) {
        if ((pid = fork()) == 0) { /* Child */
            execve("/bin/date", argv, NULL);
        }
        sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
        addjob(pid);  /* Add child to the job list */
        sigprocmask(SIG_SETMASK, &prev_all, NULL);
    }
    exit(0);
}
```

```c
void handler(int sig) {
    sigset_t mask_all, prev_all;
    pid_t pid;
    sigfillset(&mask_all);
    while ((pid = waitpid(-1, NULL, 0)) > 0) { /* Reap child */
        sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
        deletejob(pid); /* Delete child from job list */
        sigprocmask(SIG_SETMASK, &prev_all, NULL);
    }
    if (errno != ECHILD)
        unix_error("waitpid error");
}
```

Concurrency Example (3)

```c
int main(int argc, char** argv) {
    int pid;
    sigset_t mask_all, mask_one, prev_one;

    sigfillset(&mask_all);
    sigemptyset(&mask_one);
    sigaddset(&mask_one, SIGCHLD);
    Signal(SIGCHLD, handler);
    initjobs(); /* Initialize the job list */

    while (1) {
        sigprocmask(SIG_BLOCK, &mask_one, &prev_one); /* Block SIGCHLD */
        if ((pid = fork()) == 0) { /* Child process */
            sigprocmask(SIG_SETMASK, &prev_one, NULL); /* Unblock SIGCHLD */
            execve("/bin/date", argv, NULL);
        }
        sigprocmask(SIG_BLOCK, &mask_all, NULL); /* Parent process */
        addjob(pid);  /* Add the child to the job list */
        sigprocmask(SIG_SETMASK, &prev_one, NULL);  /* Unblock SIGCHLD */
    }
    exit(0);
}
```

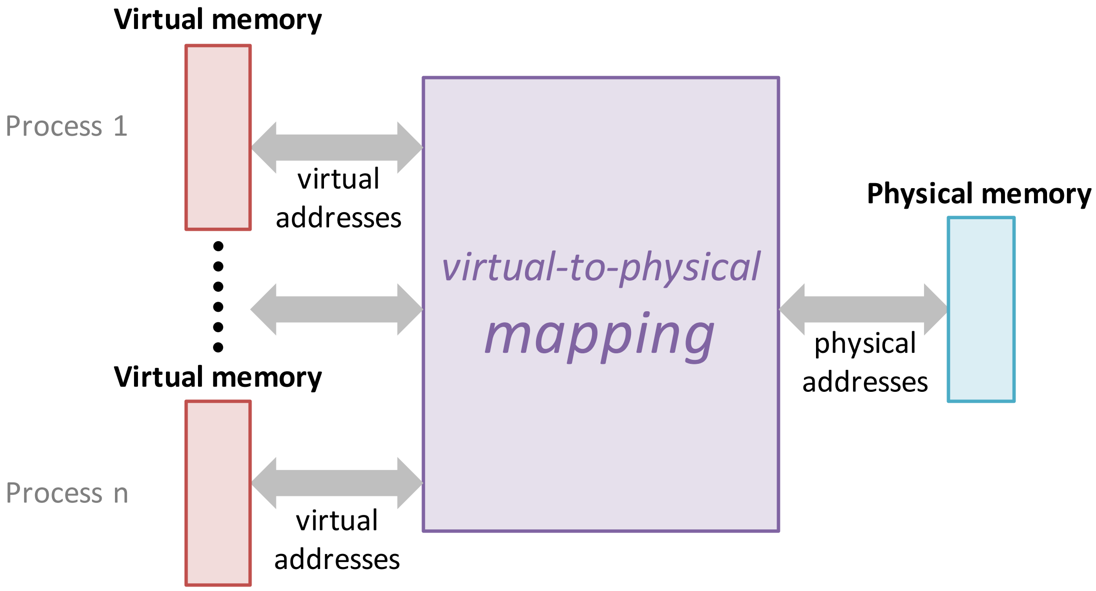
Key System Calls

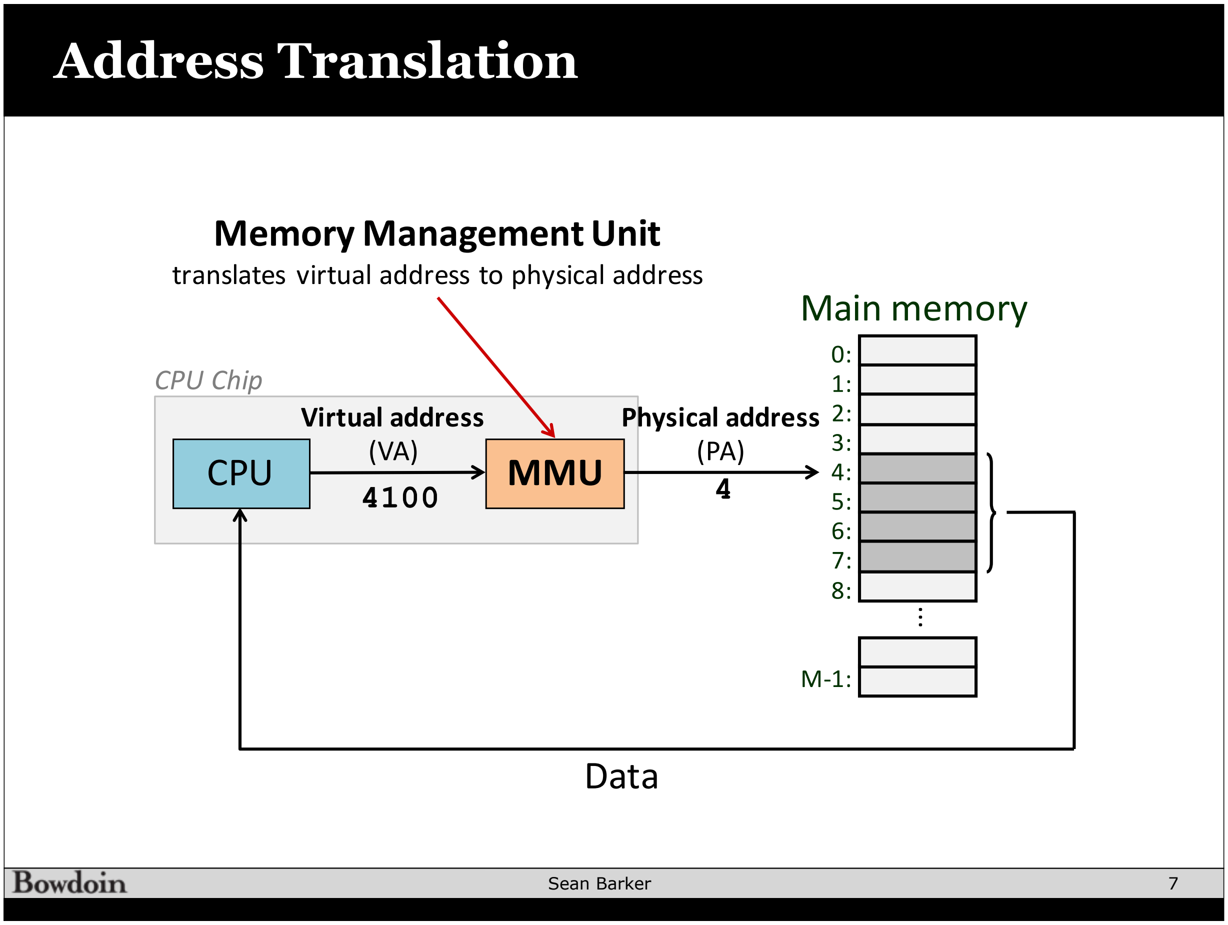
- `fork` – Create a new process
- `execve` – Run a new program

- `waitpid` – Wait for and/or reap child process
- `kill` – Send a signal
- `setpgid` – Set process group ID

- `sigprocmask` – Block or unblock signals
  - `sigemptyset` – Create empty set
  - `sigfillset` – Add every signal number to set
  - `sigaddset` – Add signal number to set
  - `sigdelset` – Delete signal number from set

Virtual Memory

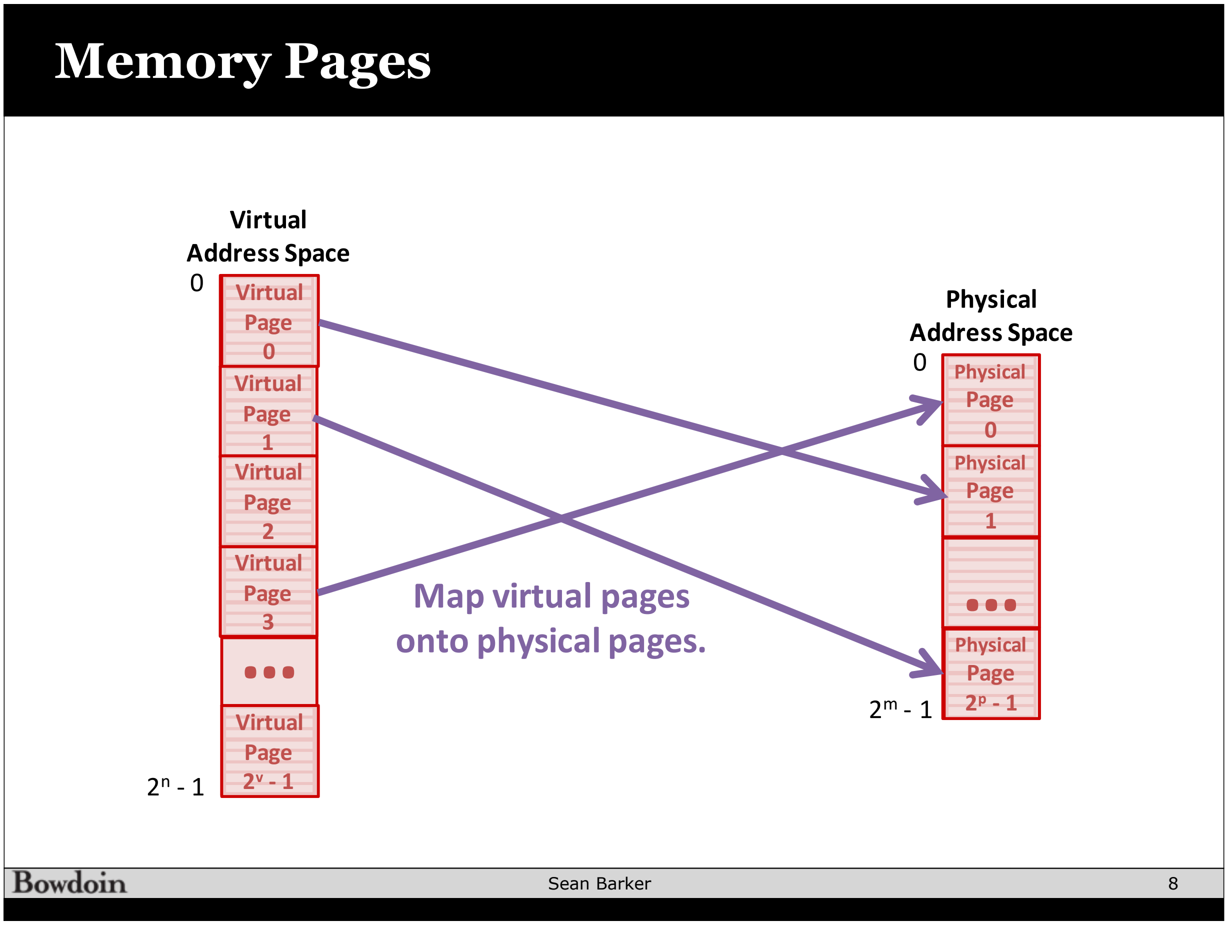




Address Translation

Memory Management Unit

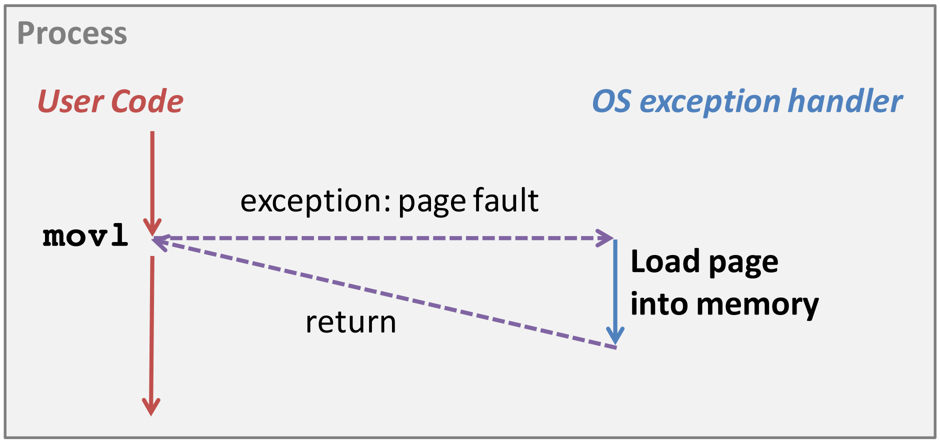
translates virtual address to physical address



Memory Pages

Map virtual pages onto physical pages.

Page Fault



Process

User Code

OS exception handler

`movl`

exception: page fault

Load page into memory

return