Introduction to C – Final part
(Finishing our UNIX shell... for now)

Announcements

- New changes in the schedule
  - Feb 02 (next Thursday):
    - Installing and running a virtual machine in VMWare (was March 01)
    - If possible, bring your laptop (backup plan)
  - Feb 07 (next Tuesday):
    - Time measurements
  - Others are shifted, except “Reading Week”
    - March 01: Mid-term exam

Processes in Foreground

```c
int child=fork();
if (child > 0) { // parent
    int status;
    wait(&status);
} else if (child == 0) {
    execvp(...)
}
```

Processes in background

- Use signals (SIGCHLD) to detect a process if finished
- Within signal handler:
  ```c
  int status=0
  int child = waitpid(-1, &status, WNOHANG);
  if (child > 0) {
      printf("Child %d terminated\n", child);
  }
  wait(&status) = waitpid(-1, &status,0)
  ```

System calls
Three approaches

- "Encapsulated" in Library calls
  ```c
  pid = getpid();
  ```

- Indirect call
  ```c
  pid = syscall(SYS_getpid);
  ```

- Inline assembly
  ```asm
  volatile asm ( "movl %eax, %0 \n\t"
              "int $0x80 \n\t":
              "=a" (pid):
              "a" (SYS_getpid));
  ```

Inline Assembly

- `asm [volatile]` (ASM_CODE:
  ```c
  [input operands]:
  [output operands]:
  [clobbers] );
  ```

- `volatile`: No optimization

- Some operands:
  - a-`eax`, b-`ebx`, c-`ecx`, d-`edx`,
  - S-`esi`, D-`edi`