last lecture: signal

for OS to tell process that something important happened

normally b/c process did something bad

signal handler examines state of the process to see what happened

SIGSEGV/SIGFPE are synchronous – handled immediately

SIGCHLD/SIGINT are asynchronous

- typically result from something external
- typically handled on a context switch (when process hops on/off CPU)
- signal handlers can be really difficult to use properly
  - can be invoked at really, really bad times (like in the middle of a malloc call or when accessing a complex data structure)
  - reentrant = program is capable of calling itself recursively
    - printf calls vsprintf, which is not reentrant – can make asynchronous signal handling very dangerous

3 steps for dealing w/ signals

1. build set sigset_t of signals we'd like to monitor
2. inform OS to suspend delivery of these signals until further notice (e.g. catalog but don't react)
   sigprocmask(SIG_BLOCK, &monitoredSignals, NULL);
   "signal process mask"
3. call sigwait (works like waitpid but for signals like processes)
   sigwait(&monitoredSet, &delivered);
   we're handling these signals <u>inline</u>! only doing 1 thing at a time

ex. Disneyland example revisited

SIGCHLD: need to be notified when children return

SIGALRM: timer that fires after x seconds as if sleep(x) is called

```
int main(→) {
    sigset_t monitored;
    buildMonitoredSet(monitored, {SIGCHLD, SIGALRM});
    blockMonitoredSignals(monitored);

    for (size_t i=0; i<5; i++) {
        pid_t pid = fork();    set of blocked signals is copied into child process! generally bad, esp. if execvp'ing
        if (pid == 0) {
            unblockMonitoredSignals(monitored);
            sleep(3*i);
            cout << ~~~~ ;
            return 0;
        }
    }
                letDadSleep(); // uses setitimer fxn (see slides for implementation)
    size_t numDone = 0;
    bool dadSeesEveryone = false;
    while (!dadSeesEveryone) {
        int delivered;
        sigwait(&monitored, &delivered);
        switch (delivered) {
            case SIGCHLD:
                numDone = reapChildProcesses(numDone);
                break;
            case SIGALRM:
                wakeupDad(numDone);
                dadSeesEveryone = numDone == 5;
```

```
                    break;
            }
        }
    }
```