Today

- Learning Outcomes
  - Use system calls that allow processes to wait on their children.

- Reading:
  - Chapter 8: 8.4

When we left our baby shell …

- You might have noticed that the parent and child thread outputs got intermingled.
- Perhaps you added a call to `dowait`, and that fixed it (or perhaps not).
- In either case, what was that `waitpid` call?

Wait and friends

- There are system calls that let a process (parent) wait for other processes (its children) to terminate.
- `pid_t wait(int *stat_loc)`
  - The calling process suspends execution until a child process terminates. Returns the `pid` of the terminated process.
  - Upon return, the `stat_loc` parameter contains information about the terminated process.
- pid_t waitpid(pid_t pid, int *stat_loc, int options)
  - `pid` indicates which process on which to wait.
  - `stat_loc` as above
  - `options`: allows either waiting for a child to terminate or checking for termination without waiting (`WNOHANG`). (And a few other less commonly used options.)

Two uses of waitpid

- In the `dowait` function of babyshell, we used `waitpid` to allow a parent to wait for its child to exit.
- Without any flag values in the option parameter, `waitpid` is a **blocking** system call.
  - That means that it stops the current process from doing any further work until something happens.
- If options specifies `WNOHANG`:
  - The call should not block if there are no processes that wish to report status.
  - That is, if there are no terminated children, return immediately.
  - This is a **nonblocking** call.

Compare and Contrast

Blocking waitpid:

Most of the time, the event on which we're waiting hasn't happened.

```
p = fork();
waitpid(p, &status, 0);
```

Polling waitpid:

The event almost always has happened.

```
p = fork();
while (p != waitpid(p, &status, WNOHANG));
```

What is the difference between these two uses?

In what case is each "better"?

Trade-offs between Blocking and Polling

- Blocking avoids wasted work.
- Blocking sometimes suffers from atomicity problems:

```
if (event hasn't happened)
    issue blocking call
```

- Polling can sometimes be more responsive:
  - If it takes you longer to block than the time it takes for the event on which you are waiting to complete, polling might be better.
- We often refer to polling as busy-waiting (we keep the processor busy while we wait). In general, you should avoid busy-waiting!