xv6©-rev10

(Copyright Frans Kaashoek, Robert Morris, and Russ Cox.)

The wait syscall

Carmi Merimovich

Tel-Aviv Academic College

February 20, 2017

sys_wait

```
3773 int sys_wait(void) {
      return wait();
     }
```

wait

- As long as we have child processes:
  - If none of the child processes is `ZOMBIE` then `sleep` on event `proc`.
  - On one of the `ZOMBIE` children do the following:
    - Release memmory: User space. kernel stack, page tables.
    - Return the `pid` of the `ZOMBIE` we just cleaned up.
- Return error since there is no child.

procfree()

```
void procfree(struct *p) {
 kfree(p->kstack);
 p->kstack = 0;
 freevm(p->pgdir);
 p->state = UNUSED;
 p->pid = 0;
 p->parent = 0;
 p->name[0] = 0;
 p->killed = 0;
}
```

wait: Look for a ZOMBIE child and free it

```
2671 int wait(void) {
      struct proc *p;
      int havekids, pid;

      acquire(&ptable.lock);
      for (;;) {
          havekids = 0;
          for (p = ptable.proc; p < &ptable.proc[NPROC]; p++)
           if (p->parent != myproc())
            continue;
           havekids = 1;
           if (p->state == ZOMBIE) {
            pid = p->pid;
            procfree(p);
            release(&ptable.lock);
            return pid;
           }
          }
```

wait: If no child error, otherwise sleep

```
        if (!havekids || myproc()->killed) {
          release(&ptable.lock);
          return -1;
        }
        sleep(myproc(), &ptable.lock);
    }
}
```