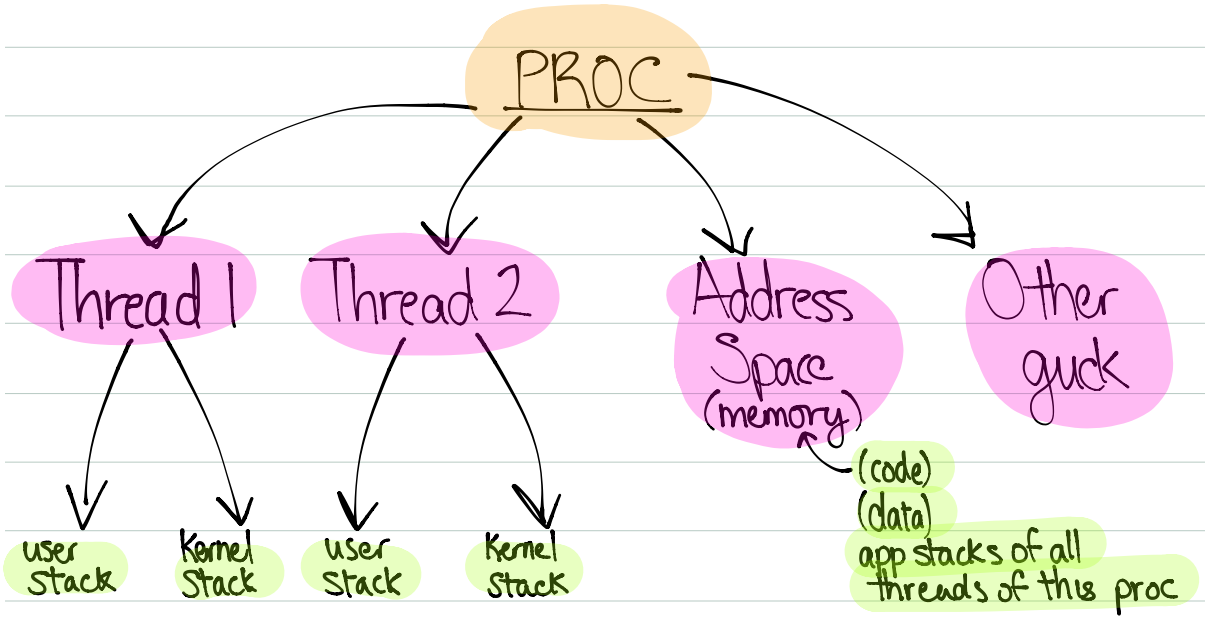

After kernel finishes booting - create first process

Then subsequent processes will be forked
↳ clone process - exact copy
→ both then return from fork
The return values will be different
- Parent will get process ID and child will return 0

waitpid - wait for one process to finish
↳ can capture the exit status of a process

ie: child_exit = wait-pid(child_pid);

if the child exits before we call waitpid - we immediately return the exit status

Therefore, it always works!

We will need to implement the following system calls

open, write, read

```
char buf[1];
fd=open("FILE", O_RDWR | O_CREATE);
rc=write(fd,"HELLO\n",6);  if(rc<0){ failed write
rc=write(fd,"WORLD\n",6);  }
lseek(fd, 0, SEEK_SET); // go to position zero
read(fd, buf, 7)  // copy from 0 to 7
```

HELLO\nWORLD\n
0 5 6

```
fd2=open("FILE", O_RDWR); // will have a different handle and offset
```

```
fd=open("FILE" ....)
rc=FORK()
 if(rc==0){
     child—read(fd)
}else{
     parent  read(fd)
}
```

For each process
-keep track of open files
  -file handle given to app
  -current file position
  -anything else...

File handles 0,1,2
-automatically available in each process

After fork()
-child has same open file handle as parent
-parent and child share current file position for each file at time of fork()
-this property is desired when you want a parent and their child to cooperate

After execv
-open files unchanged