# Network Programming

# Concurrent Programming - Processes

Note: **This class lecture will be recorded!**
If you do not consent to this recording, please do not ask questions via your video, audio or public chat; send your question to the instructor using the private chat.

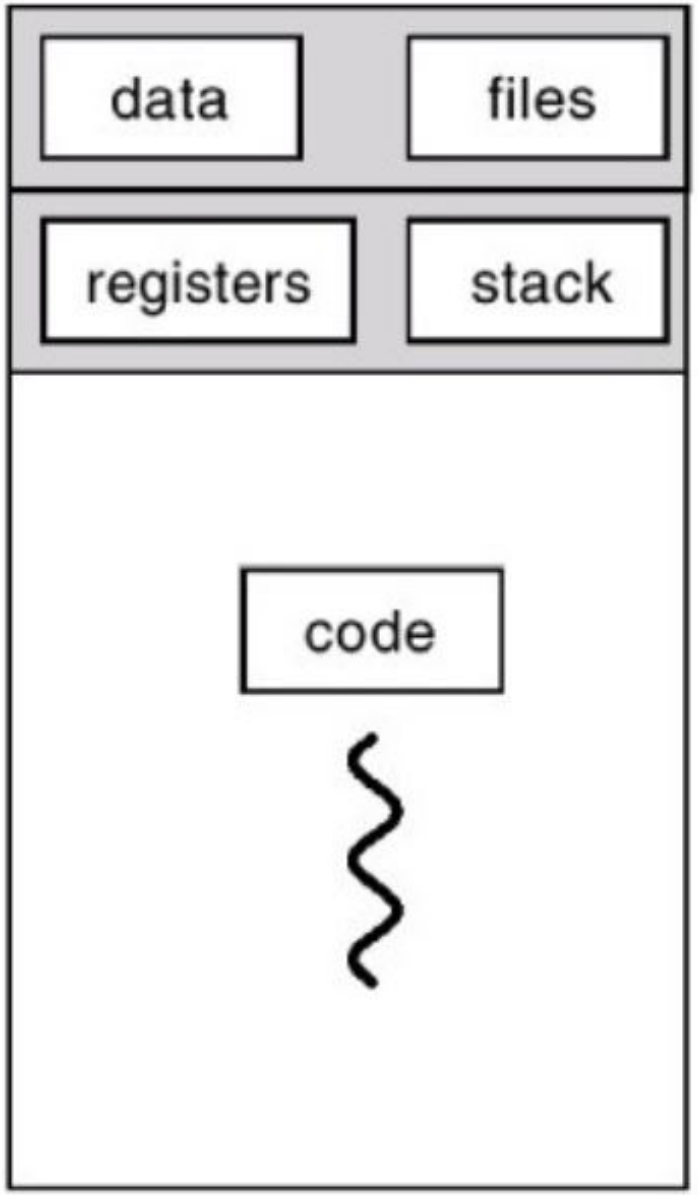
Lisa Frye, Instructor
frye@Kutztown.edu
Kutztown University

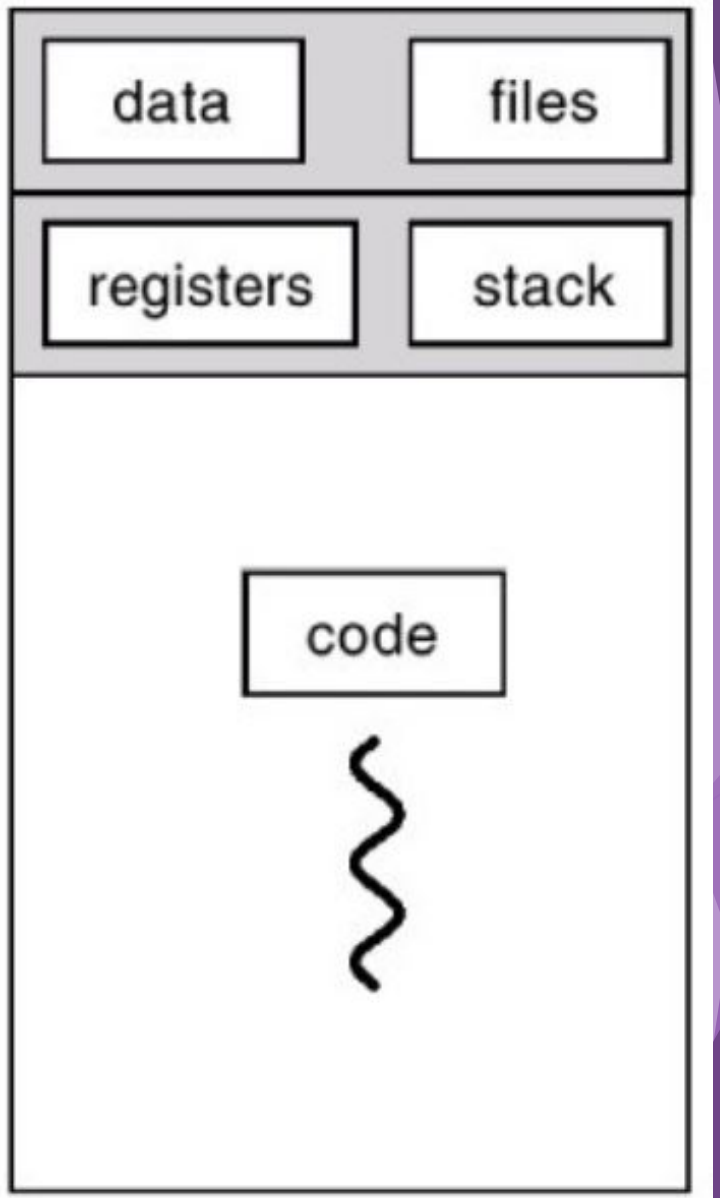
# New Process

- fork()
- Copies existing process
- Now TWO processes, so two return values!!
  - Parent – PID of child
  - Child – 0
- Data, code, stack and heap are copied**
  - Heap will depend on OS implementation; may be shared heap

# Process

```
pid_t npid;
npid = fork();
```



# Shared File Descriptors

- Child inherits parent's file descriptor table
- Shared file offsets
- Example: forkOutput.c

# OS Efficiency

- Share code area → WHY?
- Copy-On-Write (COW)
- Example

# Processes End

- *Orphan* process
- *Zombie* process

- wait()
- waitpid()
- wait3()

# New Process in Other Languages

- Python
  - fork()
  - os.wait()
- Java
  - Don’t create new processes
  - ProcessBuilder – similar
  - Fork/Join - threads

# Pointers with fork()

- Parent's address space copied
- OS implementation of heap is irrelevant

- Example
  - Value vs. address

# exec() family of calls

- Replace an existing process

- execl()
- execv()
- execle()
- execlp()
- execvp()
- execve()

- Example

- p – uses PATH
- l – list of arguments
- v – vector (array) of arguments
- e – array of environment variables