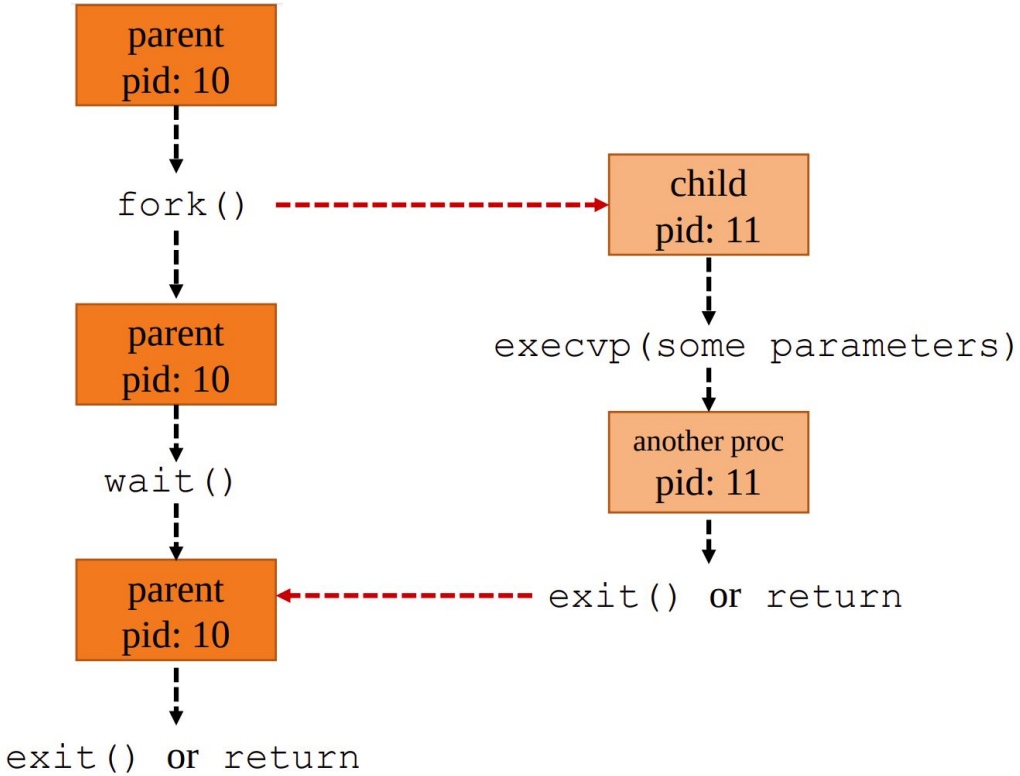
Operating System

Inter-Process Communication

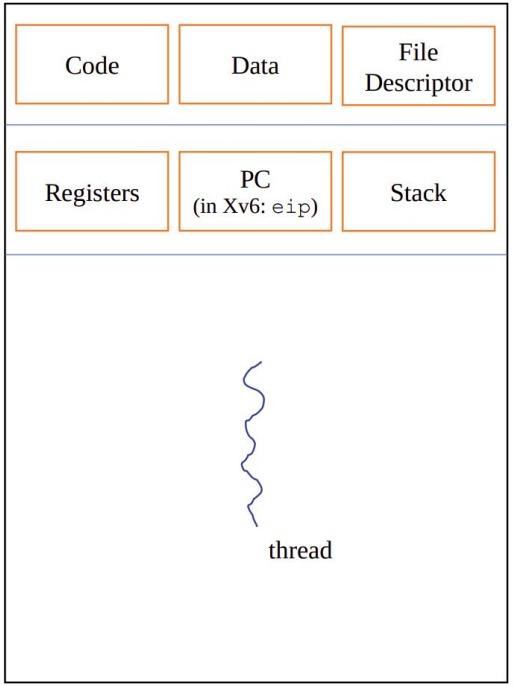
Iran University of Science and Technology

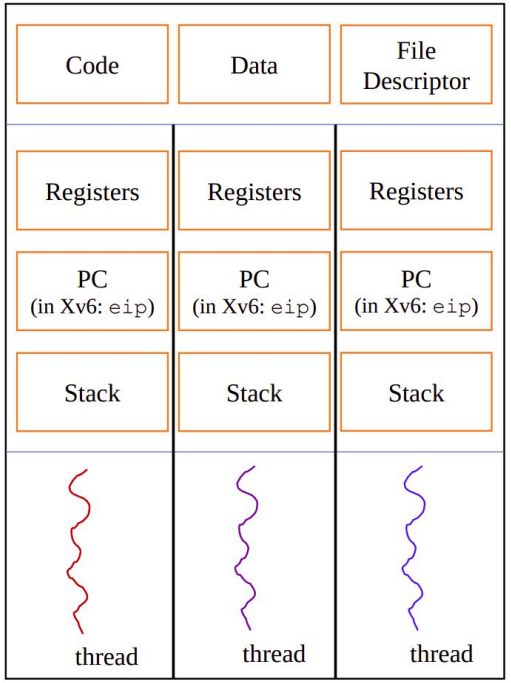
Spring 2021

fork and exec



Process and Multi-thread





Communication between threads

The code segment is shared between threads.

We can define shared variables.

All threads may access them.

The programmer should control the concurrent accesses using mutual exclusions.

How processes Communicate with each other?

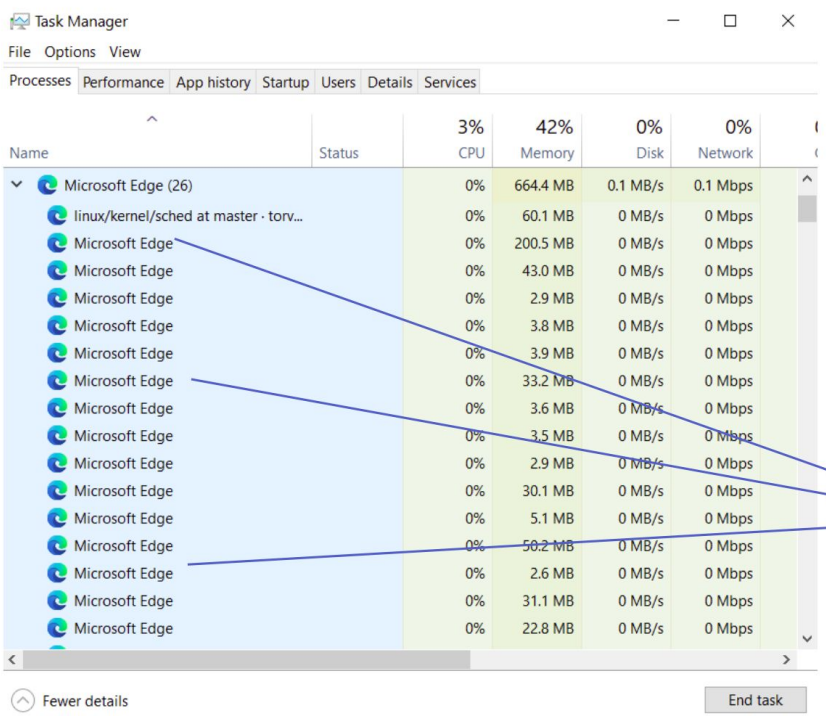
Inter-Process Communication (IPC) mechanism makes it possible for processes

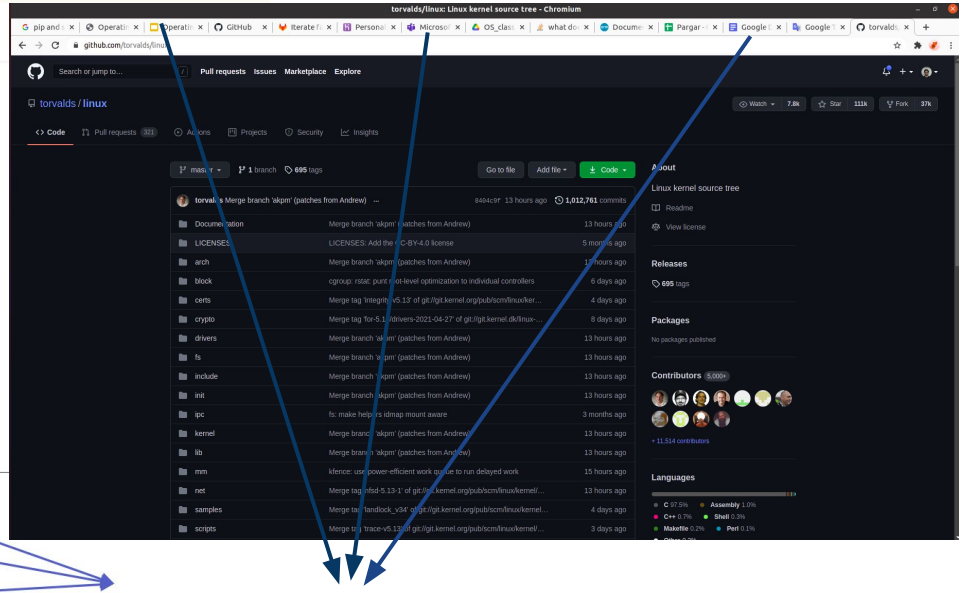
share data together.

Why?

- Information Sharing
- Computation Speed up
- Modularity

Communication between threads:

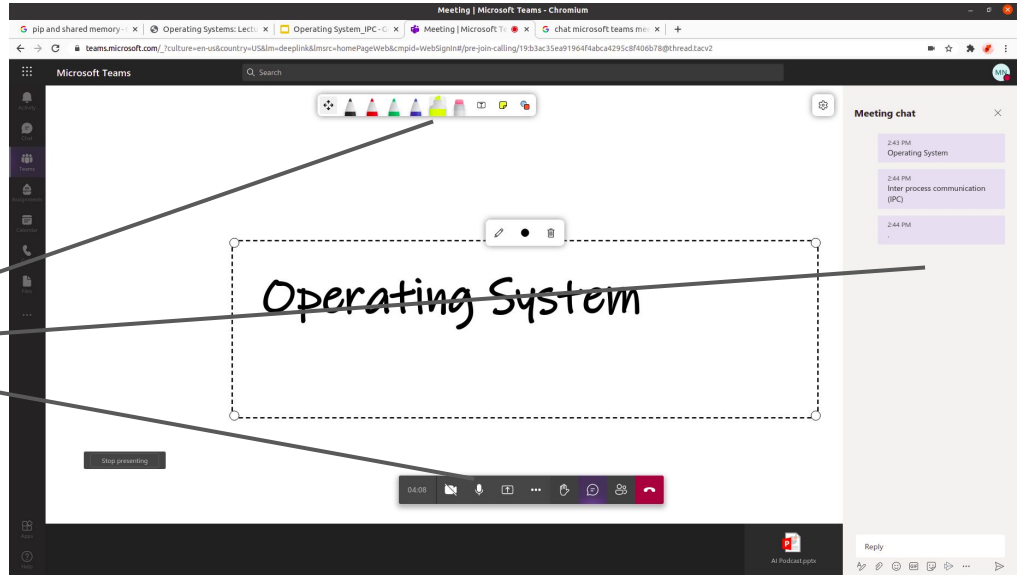




each tab represents a process

Web browser example

interactive elements within a tab could be a thread

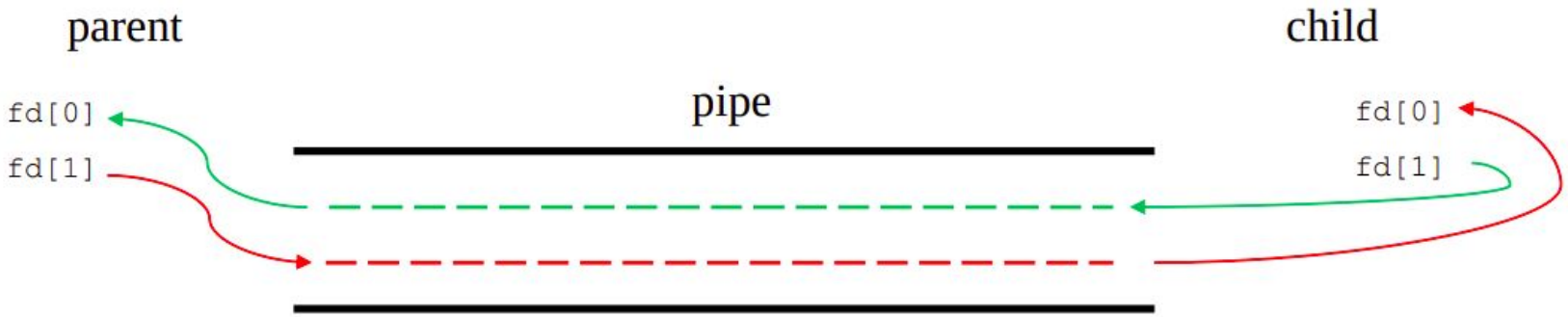


Message Passing

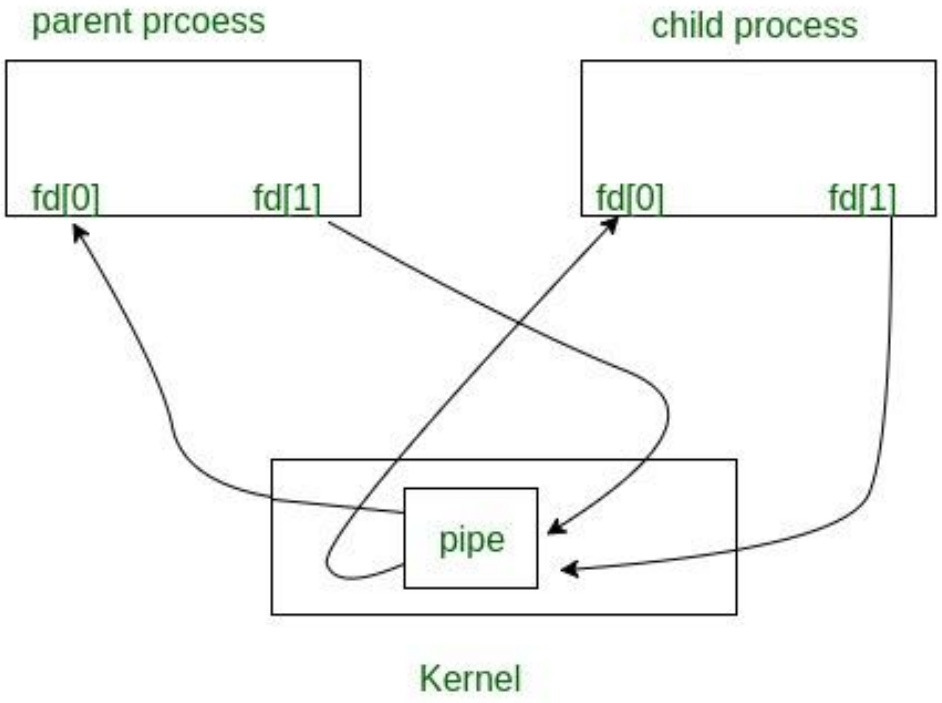
- Define variables
    - int fd[2];
    - char buff[80];
- Initialize pipe
    - pipe(fd);
- Write to and Read from pipe file descriptor
    - write(fd[1], “string”, strlen(“string”) + 1);
    - read(fd[0], buff, sizeof(buff));

Message Passing – PIPE

- I don’t agree this! But it is teached!



Message Passing – PIPE



Message Passing – PIPE

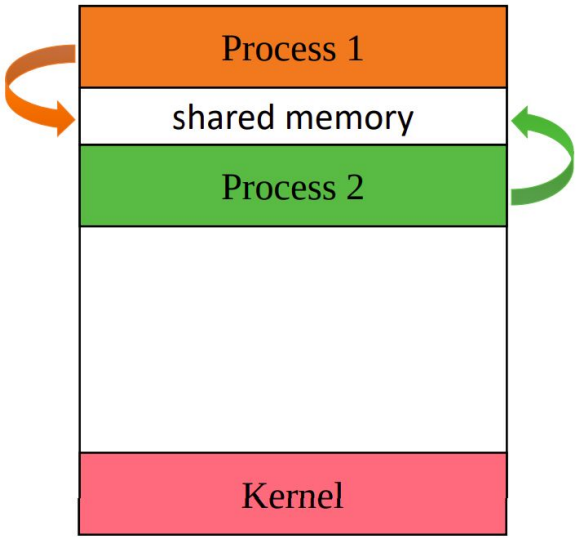
- Define file descriptor variables
  - int fd[2];
  - char buff[80];

- Initialize pipe
  - pipe(fd);

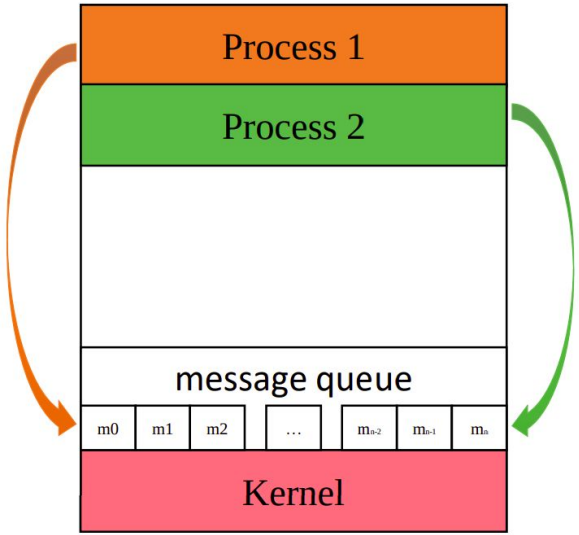
- Write to and Read from pipe file descriptor
  - write(fd[1], “string”, strlen(“string”) + 1);
  - read(fd[0], buff, sizeof(buff));

Fundamental Models

- Shared Memory



- Message Passing



Shared Memory

- Creates a region.
- This region typically resides in the address space of creator process.
- Other processes attach this segment into their address space.
- OS prevents two processes from accessing each other’s address space.
- Data exchange is not under OS control. The processes are responsible for ensuring that they are not writing data simultaneously on the same location.

Shared Memory

- int shmget(key_t key, size_t size, int shmflg)

  allocates a System Virtual shared memory segment
  input:
  key: segment identifier : number or IPC_Private(server+ cliet must child)
  size: shared segment size (in byte)
  shmflag: permission (read:S_IRUSR, write:S_IWUSR , both)

  output:
  shared memory ID

Shared Memory

- void *shmat(int shmid, const void *shmaddr, int shmflg);
  Attach to the segment to get a pointer to it.

  input:
  shmaddr: The attaching address (NuLL: OS choose adrees)
  shflag: user process permission (read or write => 0: assign both RW)

  output:
  pointer to start location of attached the segment

Questions?

END