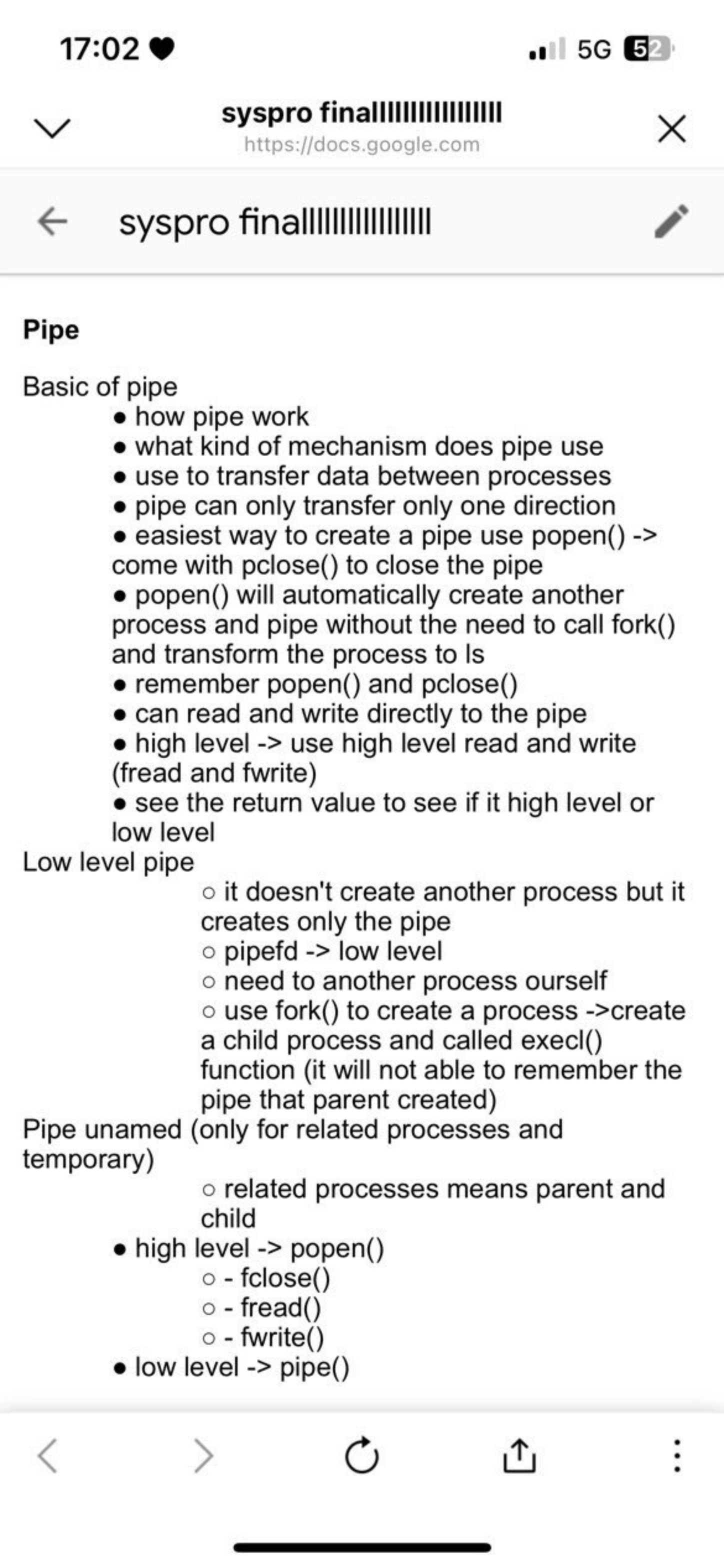
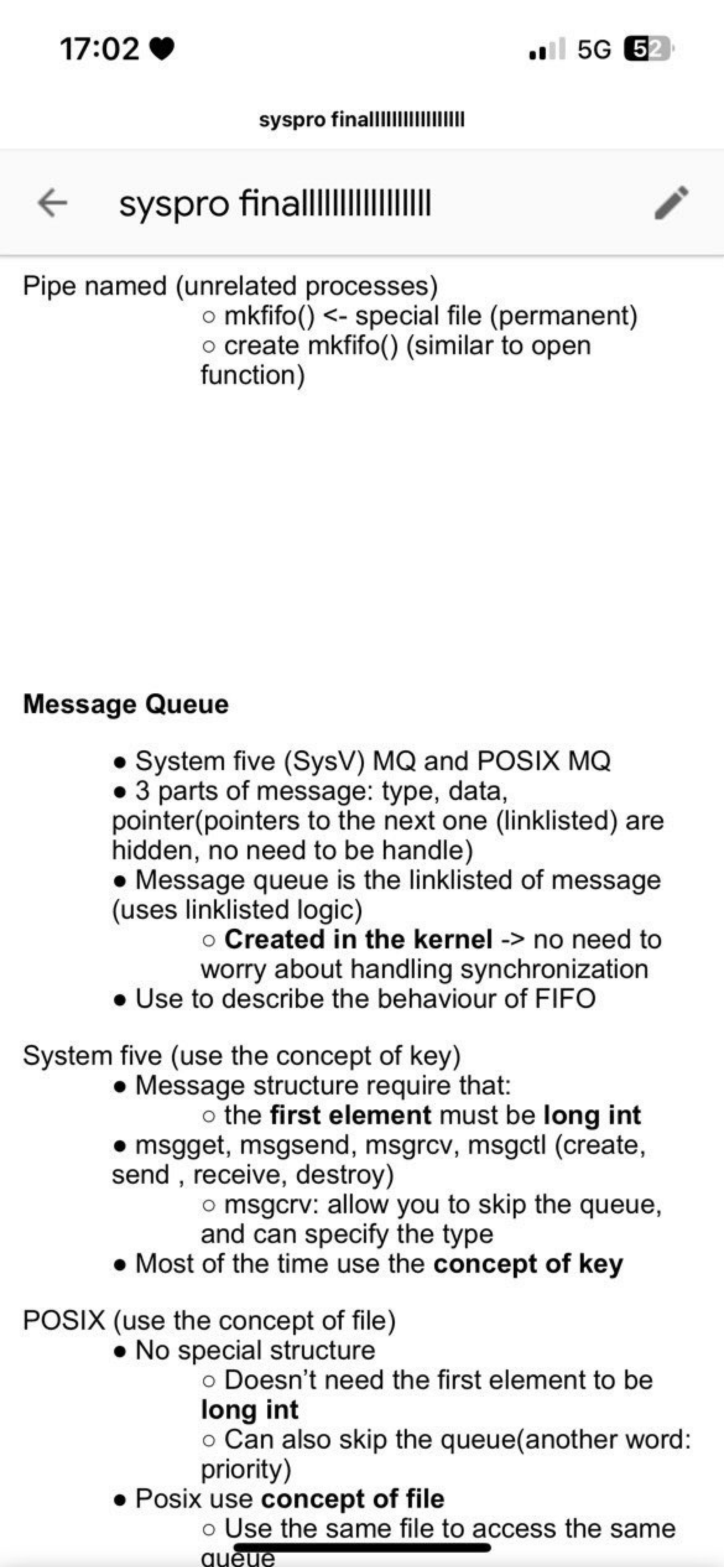


Pipe

Basic of pipe
- how pipe work
- what kind of mechanism does pipe use
- use to transfer data between processes
- pipe can only transfer only one direction
- easiest way to create a pipe use popen() -> come with pclose() to close the pipe
- popen() will automatically create another process and pipe without the need to call fork() and transform the process to ls
- remember popen() and pclose()
- can read and write directly to the pipe
- high level -> use high level read and write (fread and fwrite)
- see the return value to see if it high level or low level

Low level pipe
  - ○ it doesn't create another process but it creates only the pipe
  - ○ pipefd -> low level
  - ○ need to another process ourself
  - ○ use fork() to create a process ->create a child process and called execl() function (it will not able to remember the pipe that parent created)

Pipe unamed (only for related processes and temporary)
  - ○ related processes means parent and child
- high level -> popen()
  - ○ - fclose()
  - ○ - fread()
  - ○ - fwrite()
- low level -> pipe()



syspro finalllllllllllllllll

Pipe named (unrelated processes)

- mkfifo() <- special file (permanent)
- create mkfifo() (similar to open function)

Message Queue

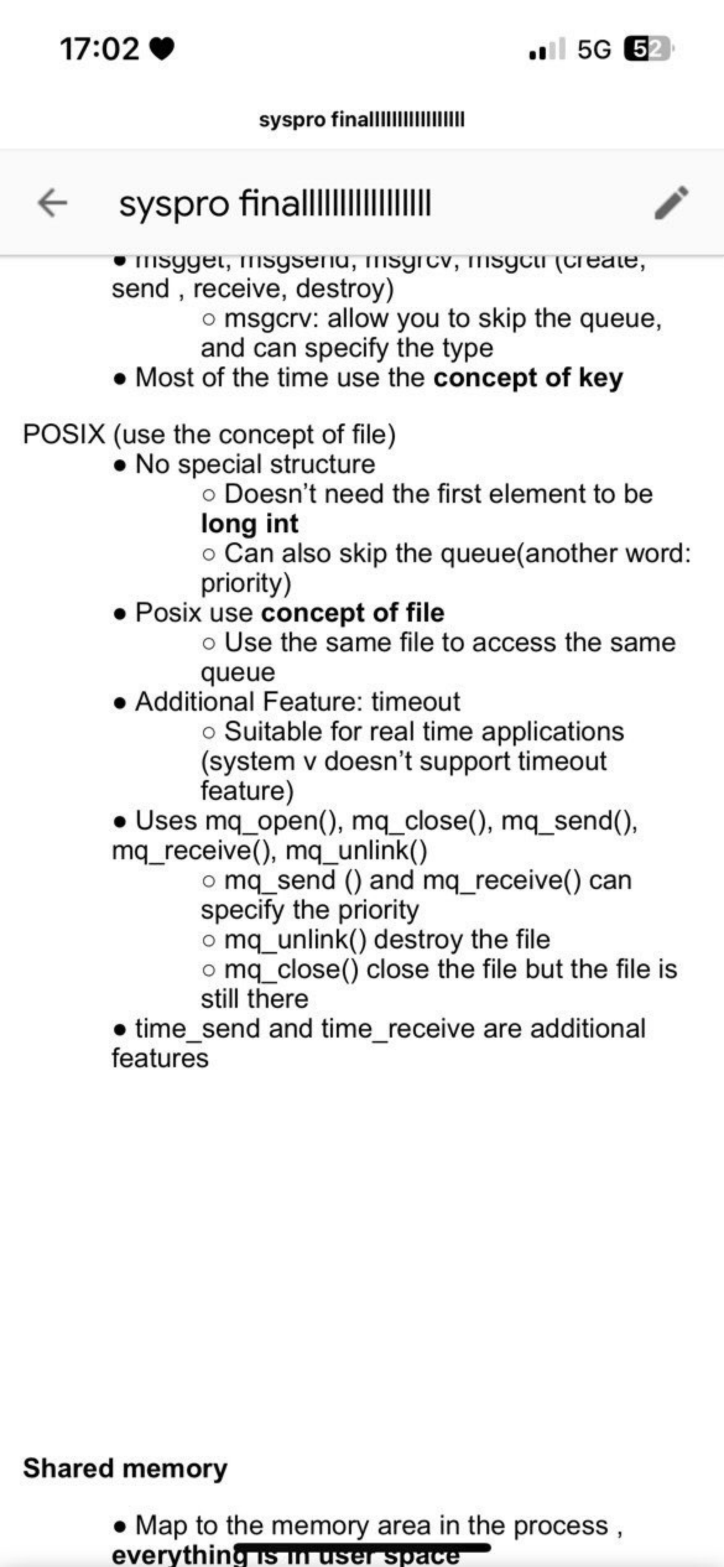
- System five (SysV) MQ and POSIX MQ
- 3 parts of message: type, data, pointer(pointers to the next one (linklisted) are hidden, no need to be handle)
- Message queue is the linklisted of message (uses linklisted logic)
    - **Created in the kernel** -> no need to worry about handling synchronization
- Use to describe the behaviour of FIFO

System five (use the concept of key)

- Message structure require that:
    - the **first element** must be **long int**
- msgget, msgsend, msgrcv, msgctl (create, send , receive, destroy)
    - msgcrv: allow you to skip the queue, and can specify the type
- Most of the time use the **concept of key**

POSIX (use the concept of file)

- No special structure
    - Doesn't need the first element to be **long int**
    - Can also skip the queue(another word: priority)
- Posix use **concept of file**
    - Use the same file to access the same queue



syspro finalllllllllllllllll

syspro finalllllllllllllllll

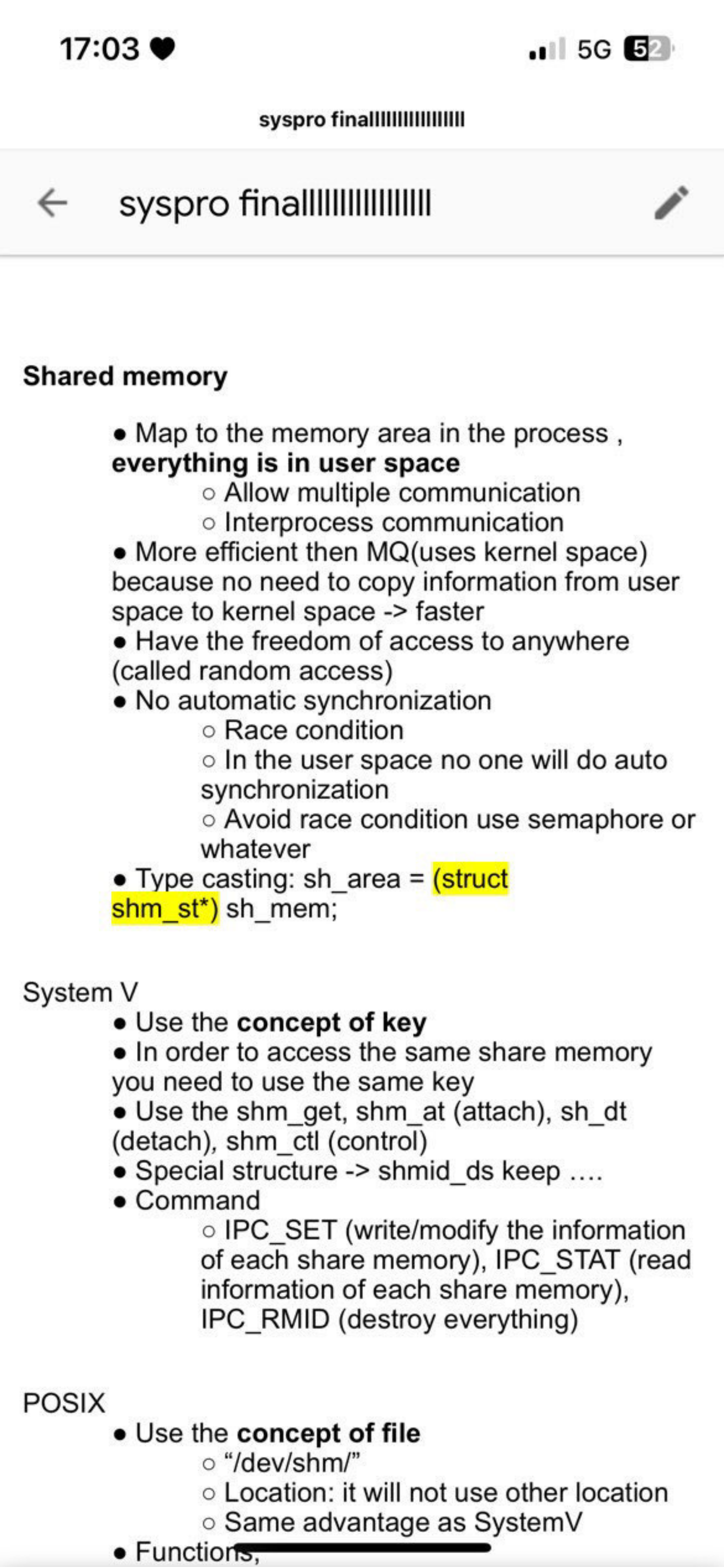
- msgget, msgsend, msgrcv, msgctl (create, send , receive, destroy)
  - msgcrv: allow you to skip the queue, and can specify the type
- Most of the time use the **concept of key**

POSIX (use the concept of file)

- No special structure
  - Doesn't need the first element to be **long int**
  - Can also skip the queue(another word: priority)
- Posix use **concept of file**
  - Use the same file to access the same queue
- Additional Feature: timeout
  - Suitable for real time applications (system v doesn't support timeout feature)
- Uses mq_open(), mq_close(), mq_send(), mq_receive(), mq_unlink()
  - mq_send () and mq_receive() can specify the priority
  - mq_unlink() destroy the file
  - mq_close() close the file but the file is still there
- time_send and time_receive are additional features

Shared memory

- Map to the memory area in the process , **everything is in user space**



Shared memory

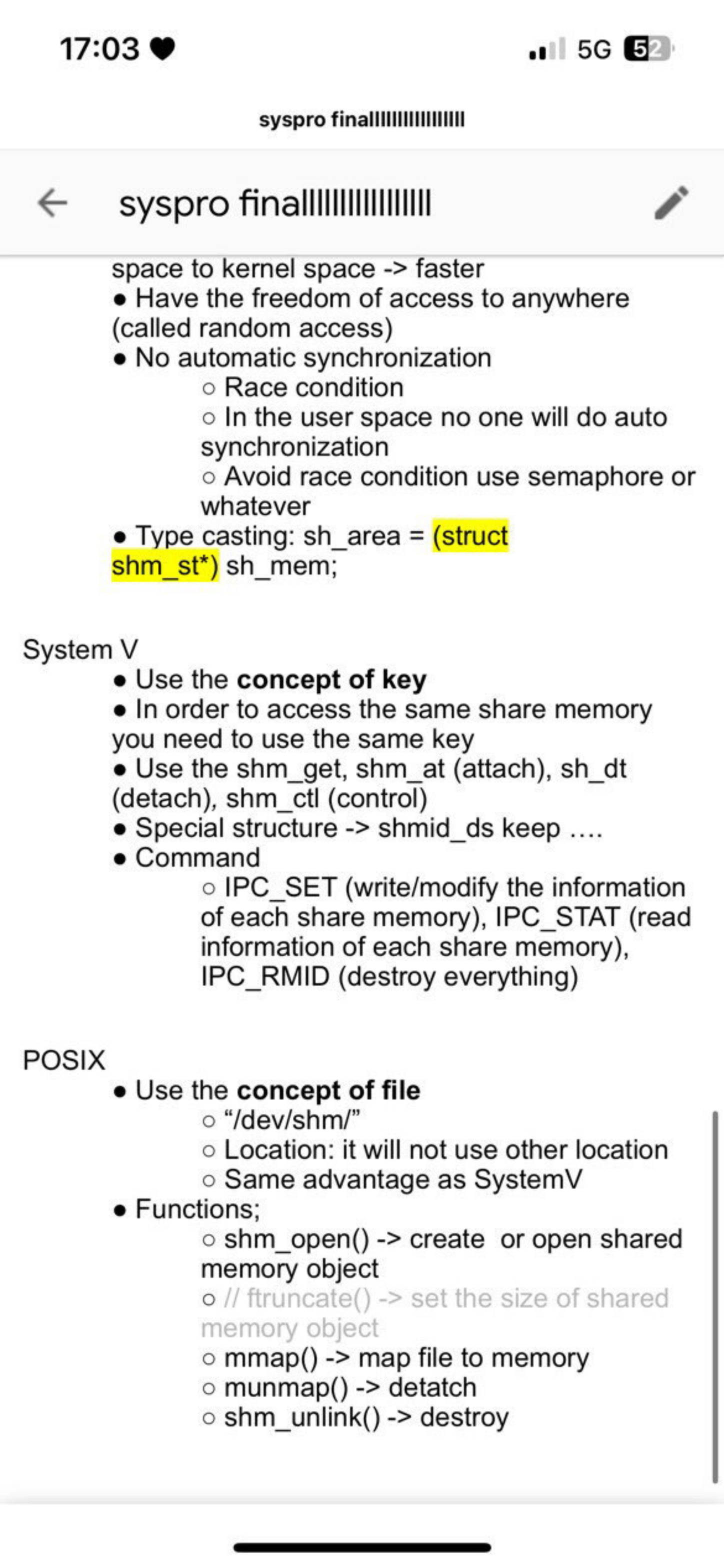
- Map to the memory area in the process , **everything is in user space**
    - Allow multiple communication
    - Interprocess communication
- More efficient then MQ(uses kernel space) because no need to copy information from user space to kernel space -> faster
- Have the freedom of access to anywhere (called random access)
- No automatic synchronization
    - Race condition
    - In the user space no one will do auto synchronization
    - Avoid race condition use semaphore or whatever
- Type casting: sh_area = (struct shm_st*) sh_mem;

System V

- Use the **concept of key**
- In order to access the same share memory you need to use the same key
- Use the shm_get, shm_at (attach), sh_dt (detach), shm_ctl (control)
- Special structure -> shmid_ds keep ….
- Command
    - IPC_SET (write/modify the information of each share memory), IPC_STAT (read information of each share memory), IPC_RMID (destroy everything)

POSIX

- Use the **concept of file**
    - “/dev/shm/”
    - Location: it will not use other location
    - Same advantage as SystemV
- Functions,



syspro finallllllllllllllllll

syspro finallllllllllllllll

space to kernel space -> faster

- Have the freedom of access to anywhere (called random access)
- No automatic synchronization
  - Race condition
  - In the user space no one will do auto synchronization
  - Avoid race condition use semaphore or whatever
- Type casting: sh_area = (struct shm_st*) sh_mem;

System V

- Use the **concept of key**
- In order to access the same share memory you need to use the same key
- Use the shm_get, shm_at (attach), sh_dt (detach), shm_ctl (control)
- Special structure -> shmid_ds keep ….
- Command
  - IPC_SET (write/modify the information of each share memory), IPC_STAT (read information of each share memory), IPC_RMID (destroy everything)

POSIX

- Use the **concept of file**
  - “/dev/shm/”
  - Location: it will not use other location
  - Same advantage as SystemV
- Functions;
  - shm_open() -> create or open shared memory object
  - // ftruncate() -> set the size of shared memory object
  - mmap() -> map file to memory
  - munmap() -> detatch
  - shm_unlink() -> destroy