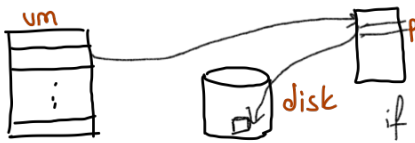
memory mapped files



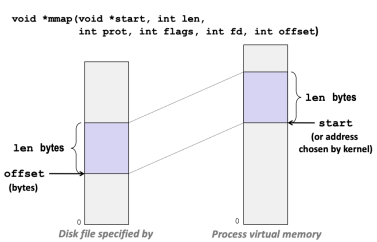
* vm areas initialized by associating them with disk objects

if you try to access the file, first page fault → copy address to virtual mem

- more flexible than shared memory → file and memory base access together
- after mapping file can be manipulated by updating the memory → no need system calls like open(), read(), write()
- the content of a file will remain even after system is shut down, nonvolatile → unlike shared memory

```
void *mmap(void *start, int len,
           int prot, int flags, int fd, int offset)
```

- **start** is the address for attachment.
  - mostly set to 0, which directs the system to choose a valid attachment address.
- **len:** The number of bytes to be attached.
  - File size should be less than or equal to this.
- **prot:** used to set the type of access (protection) for the segment such as
  - **PROT_READ, PROT_WRITE ..**
- **flags:** MAP_SHARED for a shared mapping.
  - Otherwise isolated.
  - Other options: **MAP_ANON, MAP_PRIVATE, MAP_SHARED, ...**
- **fd:** open file descriptor.
  - Once the mapping is established, the file can be closed.
- **offset:** set the starting position for the mapping.
- Return a pointer to start of mapped area (may not be **start**)




```c
#include<exchange.h>
struct Currency *curshared;
double balance = 1000;   // initial balance
int main() {
       fd = open("exchange.dat", O_RDWR | O_CREAT, 0600);
       if (fd < 0) { perror("open") ; return 1; }
       // map file as part of memory as a shared segment
       curshared = (struct Currency *) mmap(NULL,
              4*sizeof(struct Currency), PROT_READ|PROT_WRITE,
              MAP_SHARED, fd, 0);
       if (curshared == 0) { perror("mmap"); return -1;}
       close(fd);       // you can close the file afterwards
       if (curshared == NULL) return -1;
       while (fgets(line, 80, stdin)) { // trade loop
              // assume input is parsed here
              if (... "buy" ) buy(curshared+c , amount, &balance);
              if (... "sell")   sell(curshared+c , amount, &balance);
       }
       // delete the mapping
       munmap(curshared, 4*sizeof(struct Currency));
       return 0;
}
```