Assignment 2
In-memory KV-Store

By Shabir Abdul Samadh

Oct 16th/17th 2018

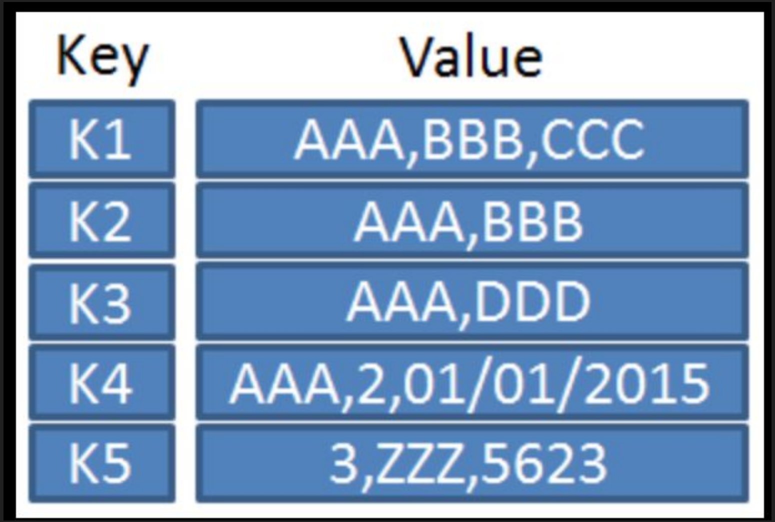
Conducted as part of the tutorial series for COMP-310 / ECSE-427 (Fall 2018)

Topics

- Key-Value store?
- Hash Functions
- General discussion/description of the assignment
    - memset()
    - memcpy()
    - Synchronization
    - Use gdb / hexedit

Key-Value Store

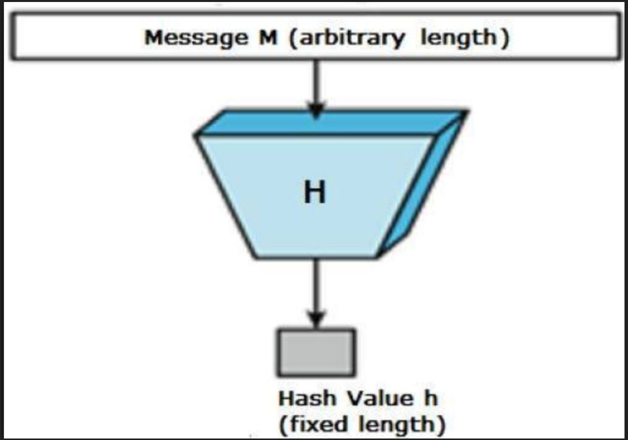
- A data storage paradigm designed for storing, retrieving, and managing associative arrays,
- Different from relational databases where the data has a specific structure
    - With KV-Store, values could be anything
- Data is treated as a single unit
- In memory KV-Store? **(eg: Redis)**
    - Kept in the system-memory (RAM)
    - Persisted until system reboot

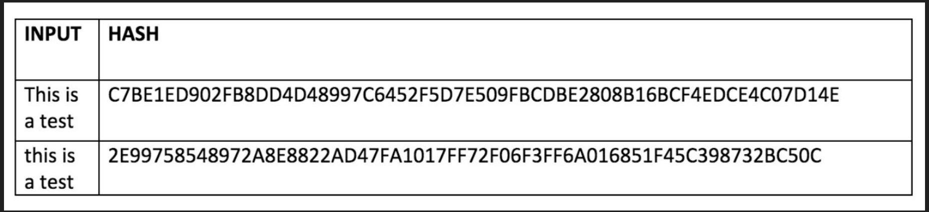


Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Hash Functions

- Any **function** that can be used to map data of arbitrary size to data of a fixed size
- The values returned by a hash function are called **hash values**, **hash codes**, **digests**, or simply **hashes**
- Always gives the same hash value for a specific data segment
- One Way functions





INPUT	HASH
This is a test	C7BE1ED902FB8DD4D48997C6452F5D7E509FBCDBE2808B16BCF4EDCE4C07D14E
this is a test	2E99758548972A8E8822AD47FA1017FF72F06F3FF6A016851F45C398732BC50C

Hash Functions

```
unsigned long hash(unsigned char *str) {
    unsigned long hash = 5381;
    int c;

    while (c = *str++)
        hash = ((hash << 5) + hash) + c;

    return (hash > 0) ? hash : -(hash);
}
```

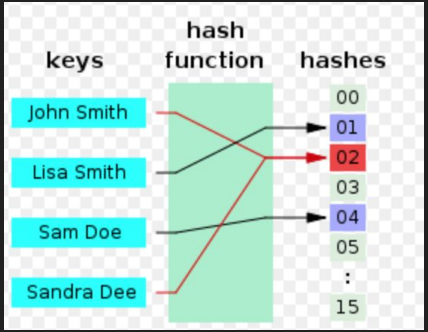
```
unsigned long hash(unsigned char *str) {
    unsigned long hash = 0;
    int c;

    while (c = *str++)
        hash= c + (hash << 6) + (hash << 16) - hash;

    return (hash > 0) ? hash : -(hash);
}
```

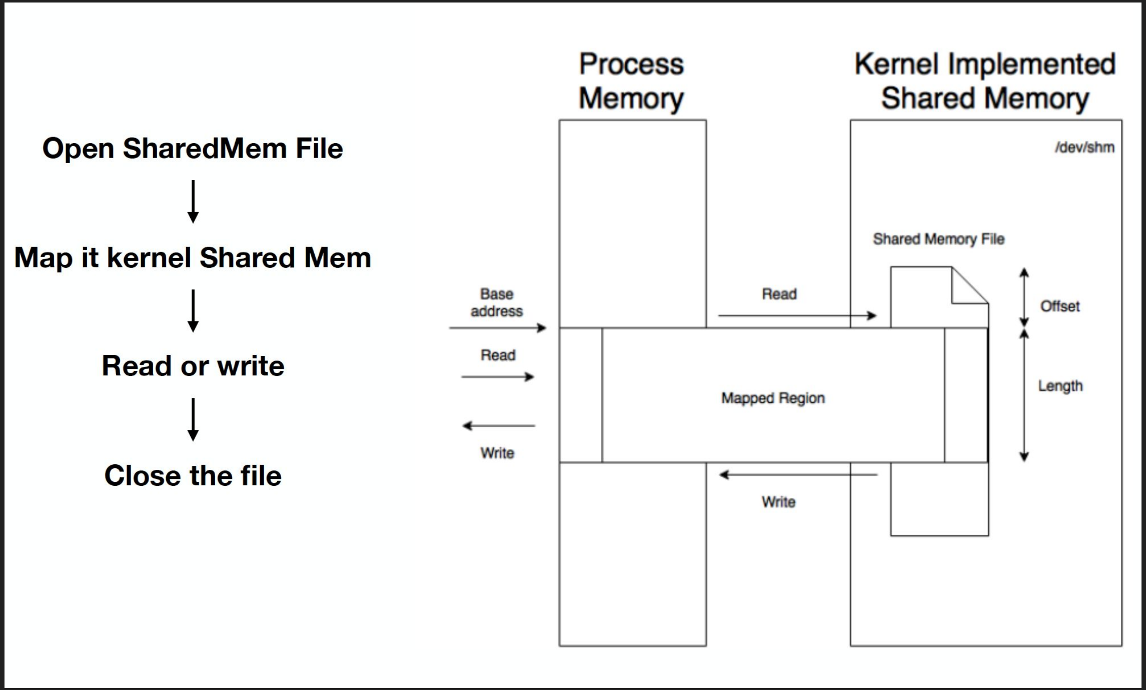
This will give a long number

You have to reduce it to the range of your KVStore



ref: http://www.cse.yorku.ca/~oz/hash.html

Shared Memory

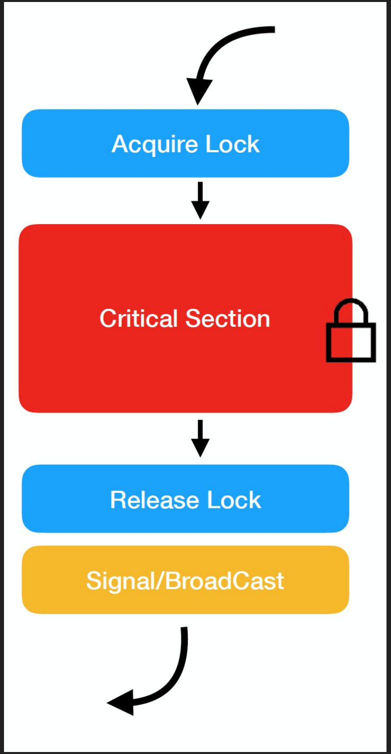


shm_open()
ftruncate()
mmap()
memcpy()
munmap()

read man page for proper syntax

created under /dev/shm

Synchronization



sem_open()

- **for creation** : requires mode and value
- **just open**

sem_wait()

sem_post()

sem_close()

sem_unlink()

Read *man page* for proper syntax

Assignment Description

```
int kv_store_create(char *kv_store_name);

      -   open shared memory segment
      -   ftruncate() to set the KVStore size
      -   mmap() to process virtual memory
      -   initialize bookkeeping information

int kv_store_write(char *key, char *value);

      -   open shared memory segment
      -   mmap() to process virtual memory
      -   calculate hash of key (if using hash)
      -   store the K-V pair in the right location
      -   many values are possible a for single key (store all of them)
```

Assignment Description

```
char *kv_store_read(char *key);

      -   open shared memory segment
      -   mmap() to process virtual memory
      -   calculate hash of the key to get the location
      -   get to location -----> read value


char **kv_store_read_all(char *key);

      -   same as above but remember each key can have multiple values
      -   read all of them and return
```

Some key points (there could be more....)

```
int kv_store_create(char *kv_store_name);

        -   check for existence of a KV-Store with same name? O_EXCL
        -   ensure all open fd(s) are close
        -   ensure to munmap()

int kv_store_write(char *key, char *value);

        -   ensure exclusive access via LOCK
        -   ensure key & value are within range (32 bytes and 256 bytes)
        -   if same key with a new value? Write key twice?
        -   if two keys map to the same location, what to do?
        -   if there is not enough space, then which value to replace
                -   How will you keep track of this value?
                -   If that is also not enough?
        -   clean up everything before leaving the function
```

Some key points (there could be more....)

```
char *kv_store_read(char *key);

    -   ensure exclusive LOCK over writers, but readers can join in
    -   check if the key exists in the KV-Store
            -   if key not found before returning release LOCKS (cleanup)
    -   if there are many records which one to return?


char **kv_store_read_all(char *key);

    -   can you call the above method inside this?
            -   what if there is a write() in between two calls?
```

Some key points (there could be more....)

General

- exclusive **LOCK** as needed (based on reader or writer)
- always remember to cleanup once done/before returning
    - free()
    - munmap()
    - sem_close() or sem_post()
    - close(fd)
- Use **calloc()** instead of **malloc()**
    - or **memset()** to ‘\0’ NULL upon **malloc()**
    - use **memcpy()** to transfer exact amounts of bytes
    - truncate strings with ‘\0’ NULL
- be cautious of the type of the pointer to memory address

Some key points (there could be more....)

- Tester requires you to implement a method to clear stuff
  - **void kv_delete_db();**
    - **sem_unlink();** -----------> all named semaphores
    - kill_shared_mem(); ------> already provided
  - name your main source file **a2_lib.c**
  - might have to add other source files to "make" if you have more

- If the program is stuck between subsequent calls - maybe semaphore was waited for and not post()'ed again.
  - remove it from **/dev/shm**
  - use **very** unique names for semaphores and KV_Store
    - comp310_james, a2_database ***(not unique!!!)***

- **cat** /dev/shm/KV_STORE > some_file
  - then view it on HexEdit to see what's on it

Synchronization

Writer

```
obtain lock(WRITE_LOCK)
write stuff
release lock(WRITE_LOCK)
```

semaphore - WRITE_LOCK
semaphore - READ_LOCK
int variable - READ_COUNT

named semaphores are system level shared

how to share read count?

Reader

```
    obtain lock(READ_LOCK)
    increment(READ_COUNT)
    If (READ_COUNT == 1)
        obtain lock(WRITE_LOCK)
    release lock(READ_LOCK)

...... readers can come in but not writers ......
    read stuff
...... readers can come in but not writers ......

    obtain lock(READ_LOCK)
    decrement(READ_COUNT)
    If (READ_COUNT == 0)
        release lock(WRITE_LOCK)
    release lock(READ_LOCK)
```

Thank You!!!

Have *patience...*

Use **gdb** to debug to check variables & memory segments...

Comment out portions of the tester and try one by one...

Open thread on **myCourses**, *email me* and I will try my best to help

Good Luck!