Test scheme - Programming Assignment 3 Part B

Binghan Geng A20482350
bgeng1@hawk.iit.edu

Yu Li A20496405
yli385@hawk.iit.edu

1. Project background

This part is to learn how to implement a pair of xv6 system calls: GetSharedPage() and FreeSharedPage() that will allow two programs (two processes) to share pages.

2. Involving platforms

Ubuntu 18.04.4 LTS + qemu + xv6 rev 9

3. Test Case

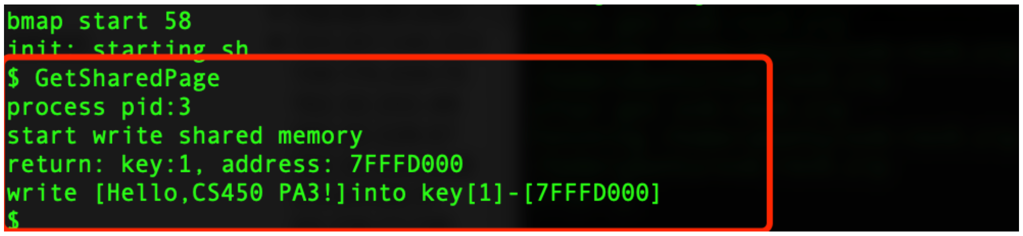
3.1 Test Case for GetSharedPage()

Execute: GetSharedPage

```
#include "types.h"
#include "stat.h"
#include "user.h"

int
main(void)
{
  printf(1,"process pid:%d \n", getpid());
  printf(1, "start write shared memory\n");

  int key, num_pages;
  key = 1;
  num_pages = 3;
  char *pa = (char*)GetSharedPage(key,num_pages);
  printf(1,"return: key:%d, address: %x \n", key, (unsigned int)pa);
  strcpy(pa, "Hello,CS450 PA3!");
  printf(1, "write [%s]into key[%d]-[%x] \n", pa, key, (unsigned int)pa);

  exit();

}
```

(GetSharedPage.c)

result:



```
bmap start 58
init: starting sh
$ GetSharedPage
process pid:3
start write shared memory
return: key:1, address: 7FFFD000
write [Hello,CS450 PA3!]into key[1]-[7FFFD000]
$
```

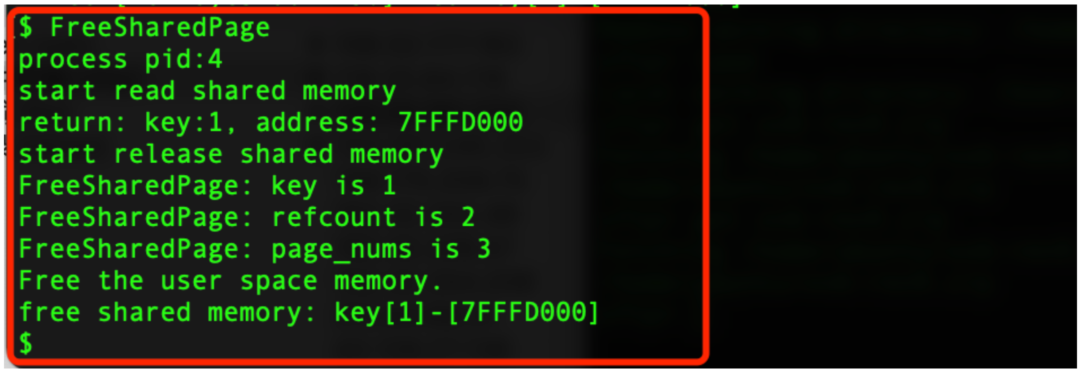
3.2 Test Case for FreeSharedPage()

Execute: FreeSharedPage

```
#include "types.h"
#include "stat.h"
#include "user.h"

int
main(void)
{
  printf(1,"process pid:%d \n", getpid());
  printf(1, "start read shared memory\n");

  int key,num_pages, r;

  key = 1;
  num_pages = 3;

  char *pa = (char*)GetSharedPage(key,num_pages);
  printf(1, "return: key:%d, address: %x \n", key, (unsigned int)pa);

  //printf(1, "read [%s] from key[%d]-[%x] \n", pa, key, (unsigned int)pa);

  printf(1, "start release shared memory\n");
  r = FreeSharedPage(key);
  if (r == -1){
    printf(1, "not shared memory free\n");
  }else {
    printf(1, "free shared memory: key[%d]-[%x] \n", key, (unsigned int) pa);
  }

  exit();
}
```

(FreeSharedPage.c)

result:

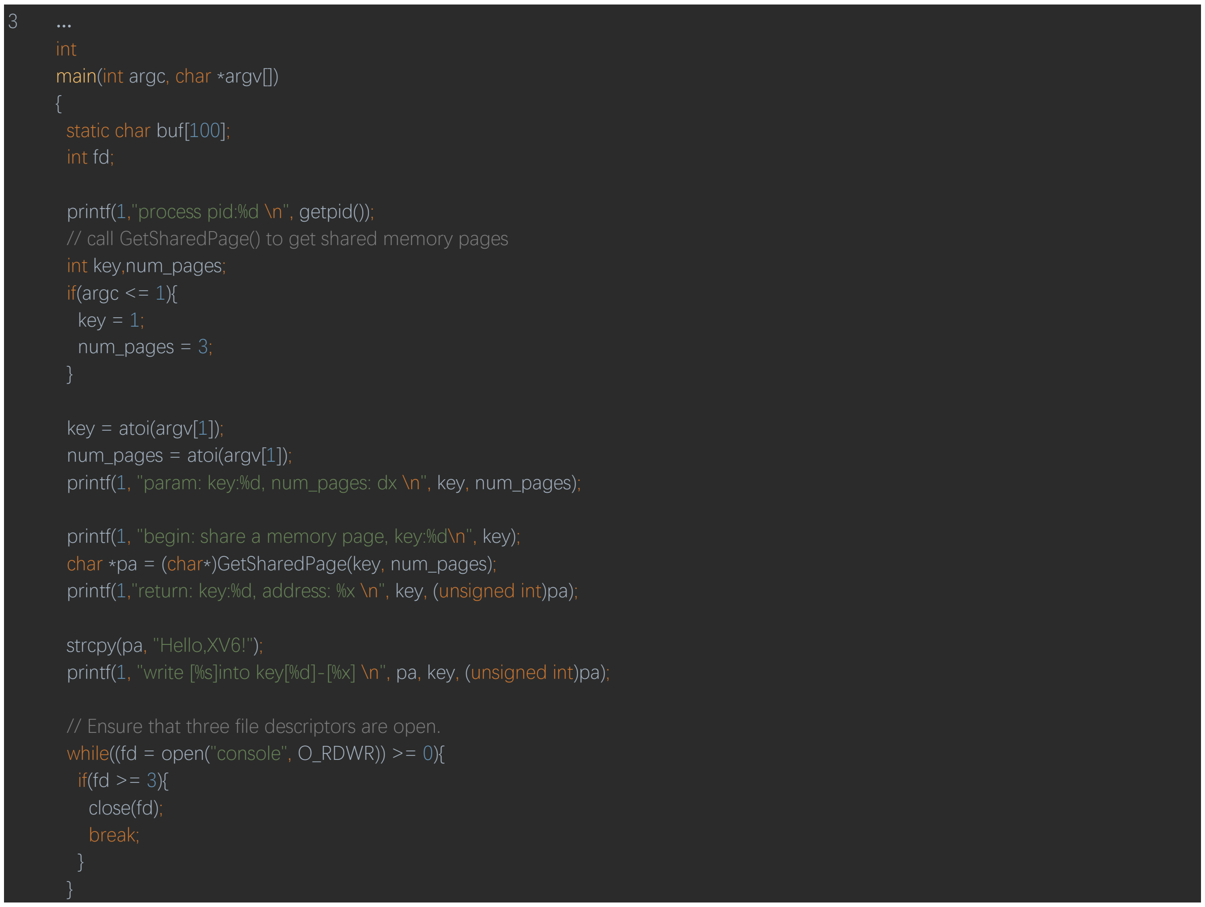


```
$ FreeSharedPage
process pid:4
start read shared memory
return: key:1, address: 7FFFD000
start release shared memory
FreeSharedPage: key is 1
FreeSharedPage: refcount is 2
FreeSharedPage: page_nums is 3
Free the user space memory.
free shared memory: key[1]-[7FFFD000]
$
```

3.3 Test Case for My_Shell (Process A)

Execute: my_shell 3 3

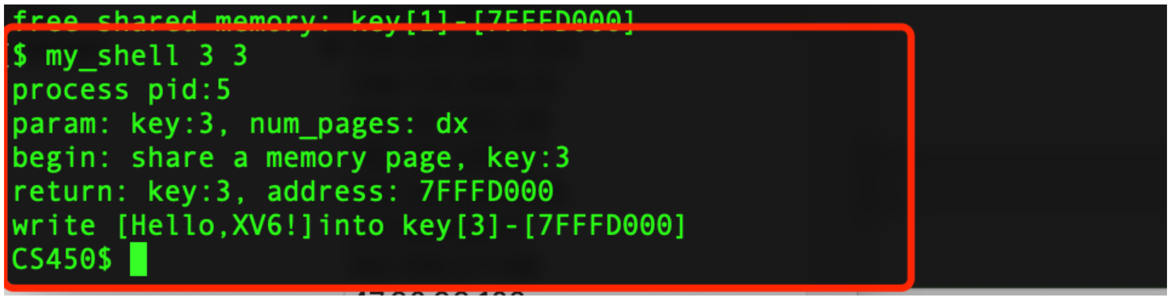
My_shell is used as the main program to call GetSharedPage according to the command line parameters to create a shared memory page， writes the initialization string “Hello, XV6!”



```
...
int
main(int argc, char *argv[])
{
 static char buf[100];
 int fd;

 printf(1,"process pid:%d \n", getpid());
 // call GetSharedPage() to get shared memory pages
 int key,num_pages;
 if(argc <= 1){
   key = 1;
   num_pages = 3;
 }

 key = atoi(argv[1]);
 num_pages = atoi(argv[1]);
 printf(1, "param: key:%d, num_pages: dx \n", key, num_pages);

 printf(1, "begin: share a memory page, key:%d\n", key);
 char *pa = (char*)GetSharedPage(key, num_pages);
 printf(1,"return: key:%d, address: %x \n", key, (unsigned int)pa);

 strcpy(pa, "Hello,XV6!");
 printf(1, "write [%s]into key[%d]-[%x] \n", pa, key, (unsigned int)pa);

 // Ensure that three file descriptors are open.
 while((fd = open("console", O_RDWR)) >= 0){
   if(fd >= 3){
     close(fd);
     break;
   }
 }
```

（my_shell.c）

result:

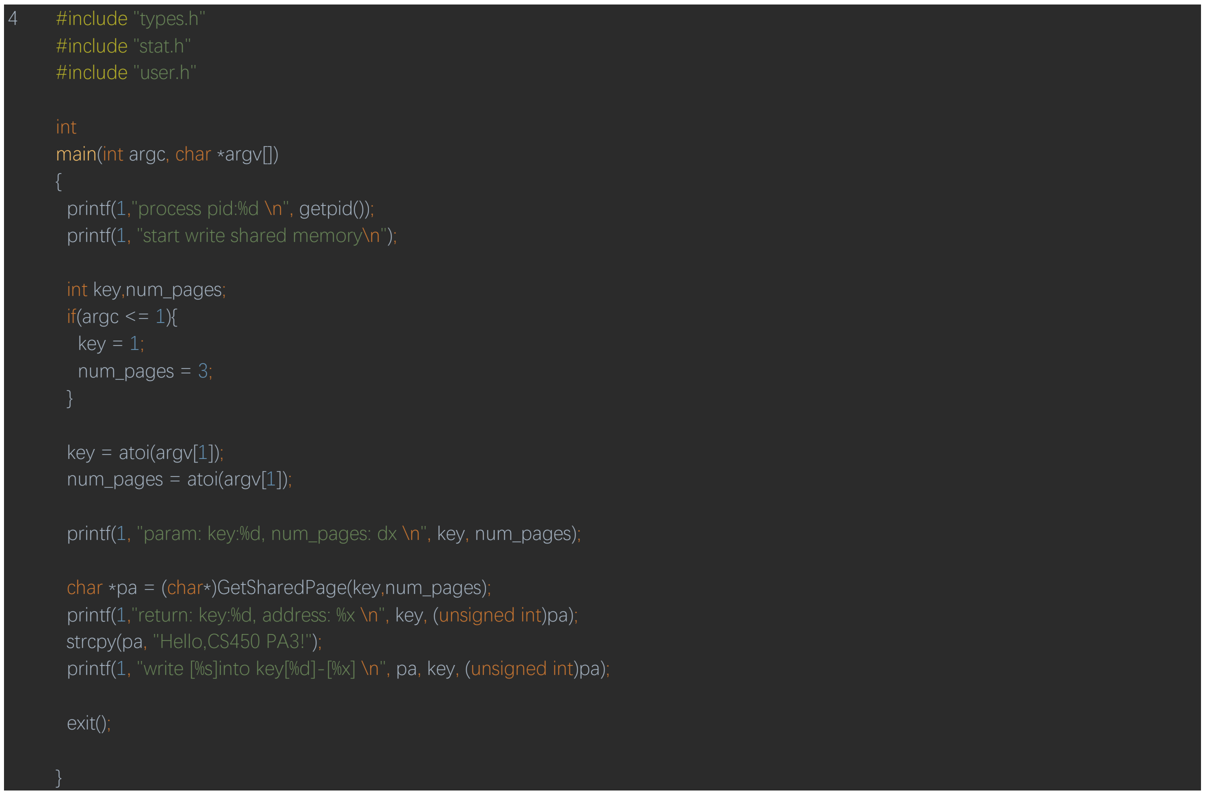


```
free shared memory: key[1]-[7FFFD000]
$ my_shell 3 3
process pid:5
param: key:3, num_pages: dx
begin: share a memory page, key:3
return: key:3, address: 7FFFD000
write [Hello,XV6!]into key[3]-[7FFFD000]
CS450$
```

3.4 Test Case for GetShmByParam (Process B)

Execute: GetShmByParam 3 3

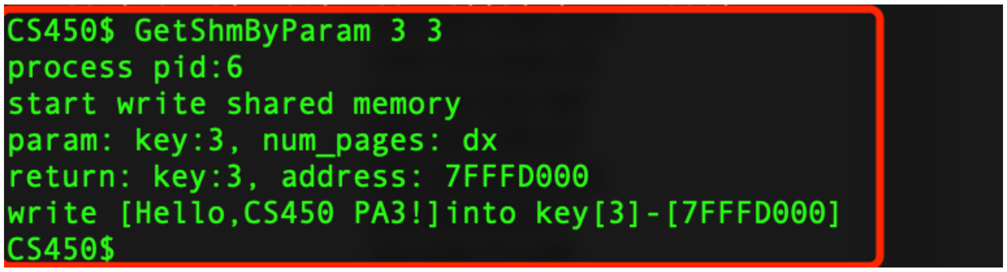
GetShmByParam obtains the shared memory address according to the Key, Read the initialization string "Hello, XV6!" written earlier, and rewrites the string "Hello,CS450 PA3!"



```c
#include "types.h"
#include "stat.h"
#include "user.h"

int
main(int argc, char *argv[])
{
  printf(1,"process pid:%d \n", getpid());
  printf(1, "start write shared memory\n");

  int key,num_pages;
  if(argc <= 1){
    key = 1;
    num_pages = 3;
  }

  key = atoi(argv[1]);
  num_pages = atoi(argv[1]);

  printf(1, "param: key:%d, num_pages: dx \n", key, num_pages);

  char *pa = (char*)GetSharedPage(key,num_pages);
  printf(1,"return: key:%d, address: %x \n", key, (unsigned int)pa);
  strcpy(pa, "Hello,CS450 PA3!");
  printf(1, "write [%s]into key[%d]-[%x] \n", pa, key, (unsigned int)pa);

  exit();

}
```

(GetShmByParam.c)

result:



```
CS450$ GetShmByParam 3 3
process pid:6
start write shared memory
param: key:3, num_pages: dx
return: key:3, address: 7FFFD000
write [Hello,CS450 PA3!]into key[3]-[7FFFD000]
CS450$
```

3.5Test Case for FreeShmByParam(Process C)

Execute: FreeShmByParam 3 3

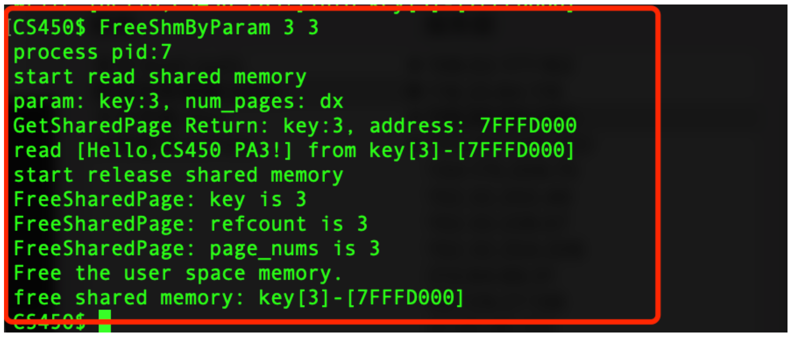
FreeShmByParam reads the string according to the shared address of the current

key=3, prints it, and then releases it.

```
#include "types.h"
#include "stat.h"
#include "user.h"

int
main(int argc, char *argv[])
{
  printf(1,"process pid:%d \n", getpid());
  printf(1, "start read shared memory\n");

  int key,num_pages, r;

  if(argc <= 1){
    key = 1;
    num_pages = 3;
  }

  key = atoi(argv[1]);
  num_pages = atoi(argv[1]);
  printf(1, "param: key:%d, num_pages: dx \n", key, num_pages);
  char *pa = (char*)GetSharedPage(key,num_pages);

  printf(1, "GetSharedPage Return: key:%d, address: %x \n", key, (unsigned int)pa);
  printf(1, "read [%s] from key[%d]-[%x] \n", pa, key, (unsigned int)pa);

  printf(1, "start release shared memory\n");
  r = FreeSharedPage(key);
  if (r == -1){
    printf(1, "not shared memory free\n");
  }else {
    printf(1, "free shared memory: key[%d]-[%x] \n", key, (unsigned int) pa);
  }

  exit();

}
```

(FreeShmByParam.c)

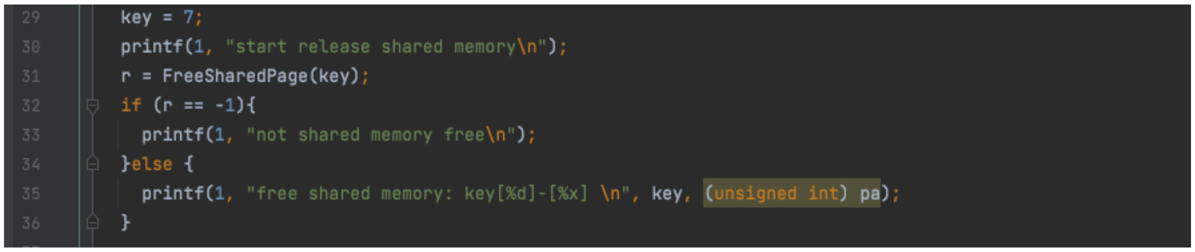
result:



```
CS450$ FreeShmByParam 3 3
process pid:7
start read shared memory
param: key:3, num_pages: dx
GetSharedPage Return: key:3, address: 7FFFD000
read [Hello,CS450 PA3!] from key[3]-[7FFFD000]
start release shared memory
FreeSharedPage: key is 3
FreeSharedPage: refcount is 3
FreeSharedPage: page_nums is 3
Free the user space memory.
free shared memory: key[3]-[7FFFD000]
CS450$
```

3.6 Test Case for FreeShmByParam(Key not exist)

Call FreeShmByParam directly, key = 7 (shared memory is not created).

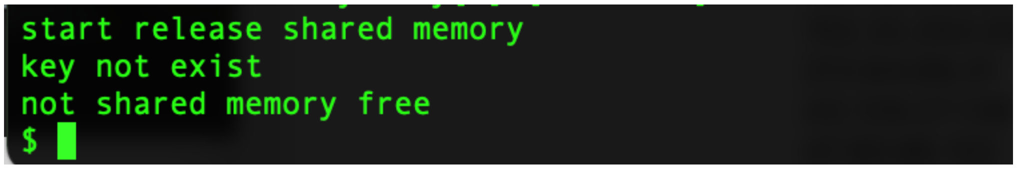


```
    key = 7;
    printf(1, "start release shared memory\n");
    r = FreeSharedPage(key);
    if (r == -1){
      printf(1, "not shared memory free\n");
    }else {
      printf(1, "free shared memory: key[%d]-[%x] \n", key, (unsigned int) pa);
    }
```

(FreeSharedPage.c)

Result:

Key not exist.



```
start release shared memory
key not exist
not shared memory free
$ 
```