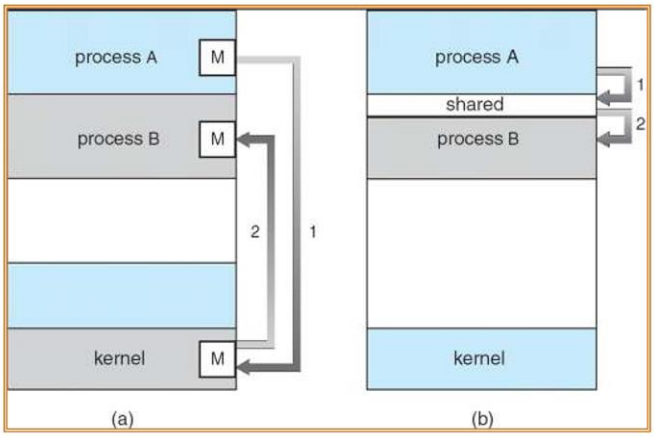
ใบงาน 8c

วัตถุประสงค์ เพื่อศึกษาวิธีการรับ/ส่งข้อมูลระหว่างโพรเซส ผ่าน shared memory บนระบบ unix

กิจกรรม

1. ศึกษา shared memory เบื้องต้น



(a) Pipe (b) Shared memory

Shared memory

- ทำงานเร็วกว่าเพราะไม่ต้อง copy ข้อมูลไปมา และไม่ต้อง switch เข้าไปใน kernel
- ต้องจัดจังหวะการรับส่งเอง

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
//Lab8c_shm1Skel.c
int main() {
  // ftok to generate unique key
  key_t key = ftok("shmfile",65);

  // shmget returns an identifier in shmid
  int shmid = shmget(key,1024,0666|IPC_CREAT);
  // shmat to attach to shared memory
  Char *str = (char*) shmat(shmid,(void*)0,0);
  //original data in shmMem
  printf("Read from mem: %s\n",str);

  //sprintf(str,"%s","os kmitl\n");
  printf("Data written to mem: %s\n",str);

  //detach from shared memory
  shmdt(str);

  // destroy the shared memory
  shmctl(shmid,IPC_RMID,NULL);

   return 0;
} //main
```

A วิธีที่ง่ายที่สุดของการหาตำแหน่งบน shrmem ที่ไม่ชนกับโพรเซสอื่นคือ ผู้พัฒนาทุกคนหา str ที่คิดว่าไม่ซ้ำมา convert ผ่าน ftok() (จะ hash ให้) เพื่อให้ client ได้ key เดียวกัน นำไปใส่ shmget() ซึ่งพารามิเตอร์ตัวที่สองมีไว้เพิ่มทางเลือกการสร้าง key เช่น นักพัฒนาใช้ path เดียวกันเป็นพารามิเตอร์ตัวแรก

B shmget() ยังต้องการขนาดและสิทธิ์ โดยฝั่ง server จะเป็นฝ่ายระบุ IPC_CREAT

C วิธีที่ง่ายที่สุดคือให้โปรแกรมมอง byte บน shrmem เป็นตัวอักษร ตอน shmat() จึง cast เป็น (char *)

D เรียก shmdt() เพื่อคืนทรัพยากร

E server เป็นฝ่ายเรียก shmctl()

4 คำสั่งมาตรฐาน การสร้าง shared memory

1. shmget() จองพื้นที่ได้ id ของ memory
2. shmat() เชื่อมตัวแปรไปที่ id พื้นที่ที่จอง
3. shmdt() ปลดการเชื่อม
4. shmctl() คืนทรัพยากร

คำสั่ง 1. เขียน output ของ ./q2

2. เขียน output ของ ./q4& และ ./q3
3. /*Q5.1*/ - /*Q5.3*/
4. ตอบ Q5.4

2. ทดสอบการสร้าง q2

เราจะสร้างโปรแกรมที่ แม่ (server) เขียน pid ไปที่ shared memory ให้ลูกอ่าน แล้วลูกเขียน pid กลับ จัดจังหวะด้วยการส่ง signal หากัน

เขียน child ก่อน
ในเมื่อยังไม่มี server ใช้ IPC_CREAT และ shmctl() ทำตัวเป็น server ไปก่อน แล้วเรียก sprint();
เราจะใช้ signal() ในการจัดจังหวะ
และ child จะเขียน os kmitl กลับให้ server อ่าน

สร้าง q2 (Lab8c_shm2Clnt.c) ด้วยการ save as จากข้อที่แล้ว

ลองรัน ./q2

```c
//Lab8c_shmb2Clnt.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include <stdlib.h>
char *str;
int isLoop = 0;
void SIGHandler(int sig) {
    signal(sig, SIG_IGN);
    printf("from handler ");
    printf("%s\n",str);

    signal(sig,SIGHandler);
}
int main() {
  signal(SIGUSR1, SIGHandler);
  // ftok to generate unique key
  key_t key = ftok("hash_this",65);

  // shmget returns an identifier in shmid
  int shmid = shmget(key,1024,0666|IPC_CREAT);

  // shmat to attach to shared memory
  str = (char*) shmat(shmid,(void*)0,0);

  while(isLoop) ;   //set to 0 to disable loop

  //str now refers to content in shmMem
  printf("Data in memory: %s\n",str);
  int ppid = atoi(str);

  //kill(SIGUSR1,ppid);
  raise(SIGUSR1);

  sprintf(str,"%s","os kmitl\n");
  printf("Writing to memory: %s\n",str);

  //detach from shared memory
  shmdt(str);

  // destroy the shared memory
  shmctl(shmid,IPC_RMID,NULL);
  return 0;
}
```

3. สร้าง q3 โดยแปลงจาก q2 (server จะเขียน pid ของตัวเองก่อน เมื่อ client อ่านแล้วส่ง SIGUSR1 ไปเพื่อแจ้งว่าอ่านเสร็จแล้ว และรอ SIGUSR1 จาก server เพื่อทำงานที่เหลือแล้วส่ง SIGUSR1 แจ้ง server เพื่อให้ server จบหลัง client (ลูกศรสีน้ำเงิน)

```c
//Lab8c_shm2Clnt.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include <stdlib.h>
char *str;
int isLoop = 0;
void SIGHandler(int sig) {
    signal(sig, SIG_IGN);
    printf("from handler ");
    printf("%s\n",str);

    signal(sig,SIGHandler);
}

int main() {
  signal(SIGUSR1, SIGHandler);
  // ftok to generate unique key

  key_t key = ftok("hash_this",65);


  // obtain shr_mem start addr
  int shmid=shmget(key,1024, 0666|IPC_CREAT);

  // shmat to attach to shared memory
  str = (char*) shmat(shmid,(void*)0,0);

  while(isLoop) ;   //skipped

  //original data in shmMem
  printf("Data in memory: %s\n",str);
  int ppid = atoi(str);

  //kill(SIGUSR1,ppid);
  raise(SIGUSR1);

  //sprintf(str,"%s","os kmitl\n");
  printf("Writing to memory: %s\n",str);

  shmdt(str); //detach from shared memory
  shmctl(shmid,IPC_RMID,NULL);
  return 0;  }
```

```c
//Lab8c_shm3Clnt.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
char *str;
int isLoop1 = 1;
void SIGHandler1(int sig) {
    signal(sig, SIG_IGN);
    //printf("from handler ");
    printf("ppid = %s,
            Btw SIGUSR1 = %d\n",str,sig);
    isLoop1 = 0;
    signal(sig,SIGHandler1);
}

int main() {
  signal(SIGUSR1, SIGHandler1);

  // ftok to generate unique key
  key_t key = ftok("hash_this",65);
  int shmid;
  // shmid = shmget(key,1024,0666|IPC_CREAT);
  shmid = shmget(key,1024,0666);

  // shmat to attach to shared memory
  str = (char*) shmat(shmid,(void*)0,0);

  //after attach to shared mem, get ppid
  int ppid = atoi(str);
  sprintf(str,"%d",getpid());
  printf("successfully obtain ppid\n");

  kill(ppid,SIGUSR1); //to server
  //raise(SIGUSR1);

  while (isLoop1) ; //waiting for signal
  printf("waiting for SIG before
                writing to server\n");

  sprintf(str,"%s","os kmitl\n");
printf("Clnt write to mem & notify: %s\n",str);
  kill(ppid,SIGUSR1); //to server

  //detach from shared memory
  shmdt(str);
  // destroy the shared memory
  //shmctl(shmid,IPC_RMID,NULL);
  return 0; } //main
```

4. สร้าง q4 (server)

```c
//Lab8c_shm3Clnt.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
char *str;

int isLoop1 = 1;
void SIGHandler1(int sig) {
  signal(sig, SIG_IGN);
  //sprintf(str,"%d",getpid()); overwrite ppid
  printf("cpid = %s,
            Btw SIGUSR1 = %d\n",str,sig);
  isLoop1 = 0;
  signal(sig,SIGHandler1); } //handler

int main() {
  signal(SIGUSR1, SIGHandler1);


  key_t key = ftok("hash_this",65);
  int shmid;
  //shmid =shmget(key,1024,0666|IPC_CREAT);
  shmid = shmget(key,1024,0666);

  // shmat to attach to shared memory
  str = (char*) shmat(shmid,(void*)0,0);

  //after attach to shared mem, get ppid
  int ppid = atoi(str);
  sprintf(str,"%d",getpid());
  printf("successfully obtain ppid\n");

  kill(ppid,SIGUSR1); //to server
  //raise(SIGUSR1);

  while (isLoop1) ; //waiting for SIGUSR1
  printf("waiting for SIG before
                writing to server\n");

  sprintf(str,"%s","os kmitl\n");
printf("Clnt write to mem & notify: %s\n",str);
  kill(ppid,SIGUSR1); //to server

  //detach from shared memory
  shmdt(str);
  // destroy the shared memory
  //shmctl(shmid,IPC_RMID,NULL);
  return 0; } //main
```

```c
//Lab8c_shmb4Serv.c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>

char *str;
int isLoop = 1;
void SIGHandler(int sig) {
   signal(sig, SIG_IGN);
   printf("from handler "); //printf("%s\n",str);
  isLoop = 0;
  signal(sig,SIGHandler); } //handler

int main(int argc, char * argv[]) {
  signal(SIGUSR1, SIGHandler);

  key_t key = ftok("hash_this",65);

  int shmid = shmget(key,1024,0666|IPC_CREAT);
  //int shmid = shmget(key,1024,0666);

  // shmat to attach to shared memory
  str = (char*) shmat(shmid,(void*)0,0);

  int cpid;
  sprintf(str,"%d",getpid()); //ppid for clnt

  //wait SIGUSR1 after child got my pid
  while(isLoop) ;
  isLoop = 1; //preparing for next loop

  cpid = atoi(str);
  kill(cpid,SIGUSR1); // to client

  while(isLoop) ;
  //reading from shmMem
  printf("From child: %s\n",str);

  //detach and remove
  shmdt(str); shmctl(shmid,IPC_RMID,NULL);
  return 0; } //main
```

ที่ prompt

./q4 & /*& จะทำให้ server ทำงาน ใน background mode กล่าวคือได้ prompt คืนมา เราสามารถ สั่ง ps -u [user] เพื่อดู pid ของ ./q4 */

./q3 /* shared memory จาก server พร้อมแล้ว*/

5. หากสามารถ fork() ได้ โครงสร้างของโปรแกรมจะเรียบขึ้นเยอะ กล่าวคือ หลังจาก fork() child จะเห็น shared memory เดียวกับ parent ไปทันที และสามารถใช้ IPC_PRIVATE ซึ่งได้ค่าที่ unique กว่า ftok

โปรแกรมนี้ count มีค่าเริ่มต้นเป็น 5 หากลูกลดค่า count และ parent เพิ่มค่า count ผลลัพธ์ที่ได้ต้องได้ 5 เท่าเดิม (หากมีการจัดจังหวะ)

โจทย์นี้ต้องการแสดงตัวอย่างการส่งข้อมูล type int โดยปรับจาก String ดังนี้ว่า เนื่องจาก String เป็น char * (อาร์เรย์ของ char) เท่ากับว่า shared memory คือ address แรกของอาร์เรย์ ดังนั้น วิธีที่ง่ายที่สุดคือมอง count เป็นอาร์เรย์ของ int โดยเรา allocate ขนาดเป็นช่องเดียว (sizeof(int))

โจทย์นี้ยังแสดงถึงสภาวะ race condition ซึ่งหมายถึง ข้อมูล (ตัวแปร) ที่ถูก(แย่งกัน)แก้ไขค่าพร้อมๆกัน โดย statement ของ โปรแกรม ซึ่งทำให้คำตอบสุดท้าย อาจไม่ใช่คำตอบที่ถูกต้อง

/* Q5.4 */ ไม่ต้องเอา wait(NULL) ออกมา ได้คำตอบ (ค่า count จาก final answer) เป็นค่าอะไรบ้าง

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <stdlib.h>
#include <wait.h>
```

```c
//Lab8c_shmb8Fork.c
int main() {

  int *count;


  int shmid;
  /* Q5.1 a or b? write the whole statement */
  /*a*/shmid = shmget(key,sizeof(int),0666|IPC_CREAT);
  /*b*/shmid = shmget(IPC_PRIVATE,sizeof(int),0666|IPC_CREAT);

  // shmat to attach to shared memory
  count = /* Q5.2 */ shmat(shmid, NULL, 0);

  count[0] = 5;

  pid_t pid;
  if ((pid = fork()) == 0) { //child
    int temp = count[0]; sleep(1); temp = temp - 1;
    /* Q5.3 */
    printf("child decrements value at %p\n", &count);
    exit(0);
  }
  //wait(NULL);
  int temp = count[0]; sleep(1); temp = temp + 1;
  count[0] = temp;
  printf("parent increments value at %p\n",&count);
  sleep(1);
  printf("final answer is %d\n",count[0]);

   //detach from shared memory
   shmdt(count);
   // destroy the shared memory
   shmctl(shmid,IPC_RMID,NULL);
  return 0;
} //main
```