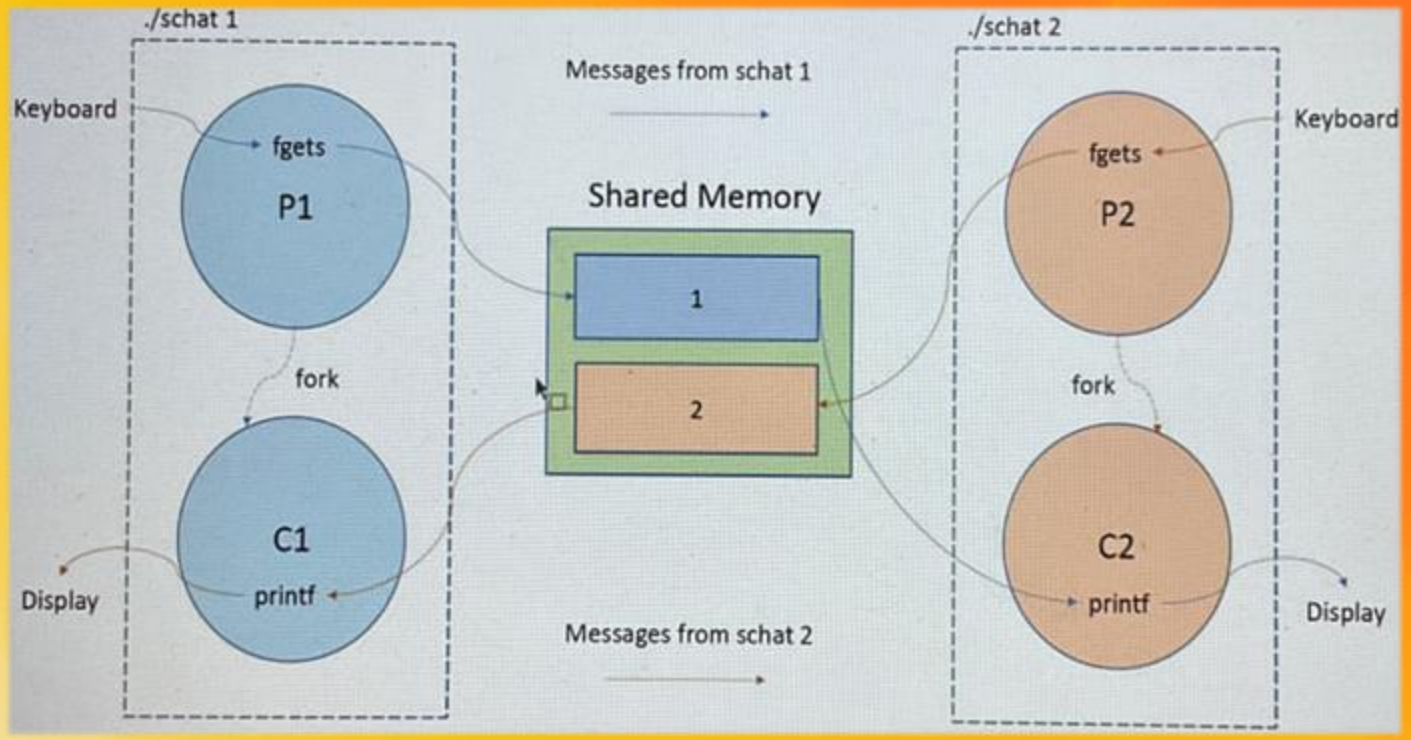
Shared Memory



```c
static void forceKill()
{
  wait();
  exit(EXIT_SUCCESS);
}
```

จะถูกเรียกโดย signal ใช้ในการแก้ zombie

```c
struct shm_st
{
  int written_0;
  char data_0[BUFSIZ];
  int written_1;
  char data_1[BUFSIZ];
};
```

สร้าง struct เก็บค่า written แสดงการ
ถูกเขียน(1) ไม่ถูกเขียน(0)
และ data เอาไว้เก็บข้อมูลของแต่ละ user

```c
if (shmdt(sh_mem) == -1 || shmctl(shmID, IPC_RMID, 0) == -1)
{
  fprintf(stderr, "shmdt or shmctl failed\n");
  exit(EXIT_FAILURE);
}
```

Detaching โดยใช้ shmdt()
ทำลายด้วย shmctl() โดย.ใช้ flag IPC_RMID

```c
int main(int argc, char* argv[])
{
  signal(SIGALRM,forceKill);

  int shmID, child;
  void* sh_mem = NULL;
  struct shm_st* sh_area;
  char buffer[BUFSIZ] = "";

  if(argc < 2)
  {
    fprintf(stderr, "Usage: %s <[1, 2]>\n", *argv);
    exit(EXIT_FAILURE);
  }

  shmID = shmget((key_t)21930, MEM_SIZE, 0666|IPC_CREAT);
  if(shmID == -1)
  {
    perror("shmget() failed\n");
    exit(EXIT_FAILURE);
  }

  sh_mem = shmat(shmID, NULL, 0);
  if (sh_mem == (void *) -1)
  {
    fprintf(stderr, "shmat failed\n");
    exit(EXIT_FAILURE);
  }
  sh_area = (struct shm_st *) sh_mem;
```

ถ้าใส่ argument ไม่ถูก จะจบโปรแกรม

รับค่า ID ของ share memory จาก shmget โดยมี key เป็น 21930 ขนาด MEM_SIZE ให้ permission read write กับ user group other ถ้าไม่มีให้สร้าง

shmat() ทำการเชื่อมเพื่อให้ processใช้ร่วมกันได้ โดยให้ระบบจัดการ(ใส่ NULL)

```
argv++;
if(strcmp(*argv, "1") == 0)
{
  child = fork();
  switch(child)
  {
    case -1: perror("Forking failed\n");
             exit(EXIT_FAILURE);
    case 0 : while(strncmp(buffer,"end chat",8))
             {
               if(sh_area->written_0)
               {
                 memset(buffer, 0, BUFSIZ);
                 strcpy(buffer, sh_area->data_0);
                 write(1,"User 2:", 7);
                 write(1, buffer, BUFSIZ);
                 sh_area->written_0=0;
               }
             }
             kill(child, SIGALRM);
             break;
```

เช็คว่าเป็นแชท 1 หรือไม่

สร้าง process ใหม่

ทำจนกว่าจะเจอ"end chat"

จะทำงาน ถ้า written_0 เป็น 1 (มีการเขียน)

อ่านค่าจาก data_0 เก็บใส่ใน buffer

แสดงทางหน้าจอ แล้วแก้ค่า written_0 ให้เป็น 0

ส่งสัญญาณให้ parent เมื่อออกลูป

```
default : while(strncmp(buffer, "end chat", 8))
        {
          memset(buffer, 0, BUFSIZ);
          int nbytes = read(0, buffer, BUFSIZ);

          if(nbytes>1)
          {
            strcpy(sh_area->data_0, buffer);
            sh_area->written_0=1;
          }
        }
        kill(child, SIGKILL);
        wait();
        break;
```

ถ้ามี SIGALRM จะให้ทำforceKill

รับค่าจาก data_0 เก็บใส่ buffer
เปลี่ยนค่า written_0 ให้เป็น 1

เมื่อออกลูปให้ส่ง SIGKILL ฆ่าลูก
แล้วรอจนลูกตายค่อยไปต่อ

```
else if (strcmp(*argv, "2") == 0)
{
  child = fork();
  switch(child)
  {
    case -1: perror("Forking failed\n");
             exit(EXIT_FAILURE);
    case 0 : while(strncmp(buffer, "end chat", 8))
             {
               if(sh_area->written_1)
               {
                 memset(buffer, 0, BUFSIZ);
                 strcpy(buffer, sh_area->data_1);
                 write(1,"User 1:", 7);
                 write(1, buffer, BUFSIZ);
                 sh_area->written_1=0;
               }
             }
             kill(child, SIGALRM);
             break;
    default : while(strncmp(buffer, "end chat", 8))
             {
               memset(buffer, 0, BUFSIZ);
               int nbytes = read(0, buffer, BUFSIZ);

               if(nbytes>1)
               {
                 strcpy(sh_area->data_0, buffer);
                 sh_area->written_0=1;
               }
             }
             kill(child, SIGKILL);
             wait();
             break;
```

ทำงานเหมือนกับแชท 1 แต่

ให้ลูกอ่านจาก data_1

ให้พ่อแม่เขียนโดยเป็น data_0