High

Performance

Distributed

System

KUAS – High Performance Distributed System

Linux Programming – IPC (Interprocess Communication)

Reporter: Po-Sen Wang

Semaphore Concept (1/3)

- 一筆資料如果同時被兩個程式寫入，會發生什麼事呢？也許是正確的結果，但更有可能是錯誤的結果。
- 一變數 A = 10 ， process1 要對變數 A 作遞增的動作， process 2 要對變數 A 作遞減的動作。得到的結果，可能是 9 、 10 、 11 其中之一，然而正確的答案是 10 。
- 所以，要避免一筆資料同時被寫入，我們需要 semaphore 來限制能進行寫入的程式。

Semaphore Concept (2/3)

- 最簡單的 semaphore 是一個只能有 0 或 1 的變數，它利用兩個操作 P 和 V 來控制。P 與 V 的定義如下：
  - P：若 semaphore 大於 0 則 semaphore 減少 1
    ，若 semaphore 為 0 則暫停程序。
  - V：將 semaphore 增加 1，並恢復其他被暫停的
    程序。

Semaphore Concept (3/3)

- Semaphore 的虛擬碼：

```
semaphore sv = 1;

loop forever {
  P(sv);
  critical code section;
  V(sv);
  non-critical code section;
}
```

Semaphore Function (1/4)

- #include <sys/sem.h>
- #include <sys/type.h>
- #include <sys/ipc.h>

- int semget(key_t *key*, int *num_sems*, int *sem_flags*);

  建立新的 semaphore 或取得現有的 semaphore id 。
- int semop(int *sem_id*, struct sembuf **sem_ops*, size_t *num_sem_ops*);

  允許對 semaphore 資訊的直接存取。
- int semctl(int *sem_id*, int *sem_num*, int *command*, union semun *sem_union*);

  用來改變 semaphore 值。

Semaphore Function (2/4)

- int semget(key_t *key*, int *num_sems*, int *sem_flags*);
  - *key*: 用來允許不相關的程序存取相同的 semaphore 。
  - *num_sems*: 需要的 semaphore 數，通常為 1 。
  - *sem_flags*: Semaphore 的權限。在建立新的 semaphore 時要加上 IPC_CREATE 。

Semaphore Function (3/4)

- int semop(int *sem_id*, struct sembuf **sem_ops*, size_t *num_sem_ops*);
  - *sem_id*: Semaphore id.
  - *sem_ops*:

```
struct sembuf {
  short sem_num; // 通常設為 0，除非是使用 semaphore 陣列。
  short sem_op; // 設為 -1 代表 semaphore 的 P 動作；設為 +1 代
                                表 semaphore 的 V 動作。
  short sem_flg; // 一般設定為 SEM_UNDO。
}
```

  - *num_sem_ops*: *sem_ops* 的數量。

Semaphore Function (4/4)

- int semctl(int *sem_id*, int *sem_num*, int *command*, union semun *sem_union*);
  - *sem_id*: Semaphore id.
  - *sem_num*: 通常為 0 。
  - *command*: SETVAL ：用來將 semaphore 起始化。
    IPC_RMID ：用來刪除 semaphore 。
  - union semun {

```
    int val; /* Value for SETVAL */
    struct semid_ds *buf; /* Buffer for IPC_STAT, IPC_SET */
    unsigned short *array; /* Array for GETALL, SETALL */
    struct seminfo *__buf; /* Buffer for IPC_INFO */
};
```

Semaphore Example (1/6)

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#include "semun.h"

static int set_semvalue(void);
static void del_semvalue(void);
static int semaphore_p(void);
static int semaphore_v(void);

static int sem_id;
```

Semaphore Example (2/6)

```
int main(int argc, char *argv[])
{
    int i;
    int pause_time;
    char *op_char_in = "process2 in";
    char *op_char_out = "process2 out";

    srand((unsigned int)getpid());

    sem_id = semget((key_t)888, 1, 0666 | IPC_CREAT);

    if (argc > 1) {
        if (!set_semvalue()) {
            fprintf(stderr, "Failed to initialize semaphore\n");
            exit(EXIT_FAILURE);
        }
        op_char_in = "process1 in";
        op_char_out = "process1 out";
        sleep(2);
    }
```

Semaphore Example (3/6)

```
    for(i = 0; i < 5; i++) {

        if (!semaphore_p()) exit(EXIT_FAILURE);
        printf("%s\t", op_char_in);fflush(stdout);
        pause_time = rand() % 3;
        sleep(pause_time);
        printf("%s\n", op_char_out);fflush(stdout);

        if (!semaphore_v()) exit(EXIT_FAILURE);

        pause_time = rand() % 2;
        sleep(pause_time);
    }

    printf("\n%d - finished\n", getpid());

    if (argc > 1) {
        sleep(10);
        del_semvalue();
    }

    exit(EXIT_SUCCESS);
}
```

Semaphore Example (4/6)

```
static int set_semvalue(void)
{
    union semun sem_union;

    sem_union.val = 1;
    if (semctl(sem_id, 0, SETVAL, sem_union) == -1) return(0);
    return(1);
}

static void del_semvalue(void)
{
    union semun sem_union;

    if (semctl(sem_id, 0, IPC_RMID, sem_union) == -1)
        fprintf(stderr, "Failed to delete semaphore\n");
}
```

Semaphore Example (5/6)

```
static int semaphore_p(void)
{
    struct sembuf sem_b;

    sem_b.sem_num = 0;
    sem_b.sem_op = -1; /* P() */
    sem_b.sem_flg = SEM_UNDO;
    if (semop(sem_id, &sem_b, 1) == -1) {
        fprintf(stderr, "semaphore_p failed\n");
        return(0);
    }
    return(1);
}
```

```
static int semaphore_v(void)
{
    struct sembuf sem_b;

    sem_b.sem_num = 0;
    sem_b.sem_op = 1; /* V() */
    sem_b.sem_flg = SEM_UNDO;
    if (semop(sem_id, &sem_b, 1) == -1) {
        fprintf(stderr, "semaphore_v failed\n");
        return(0);
    }
    return(1);
}
```

Semaphore Example (6/6)

kevin@localhost:~/linux_program/ch13 [80x24]

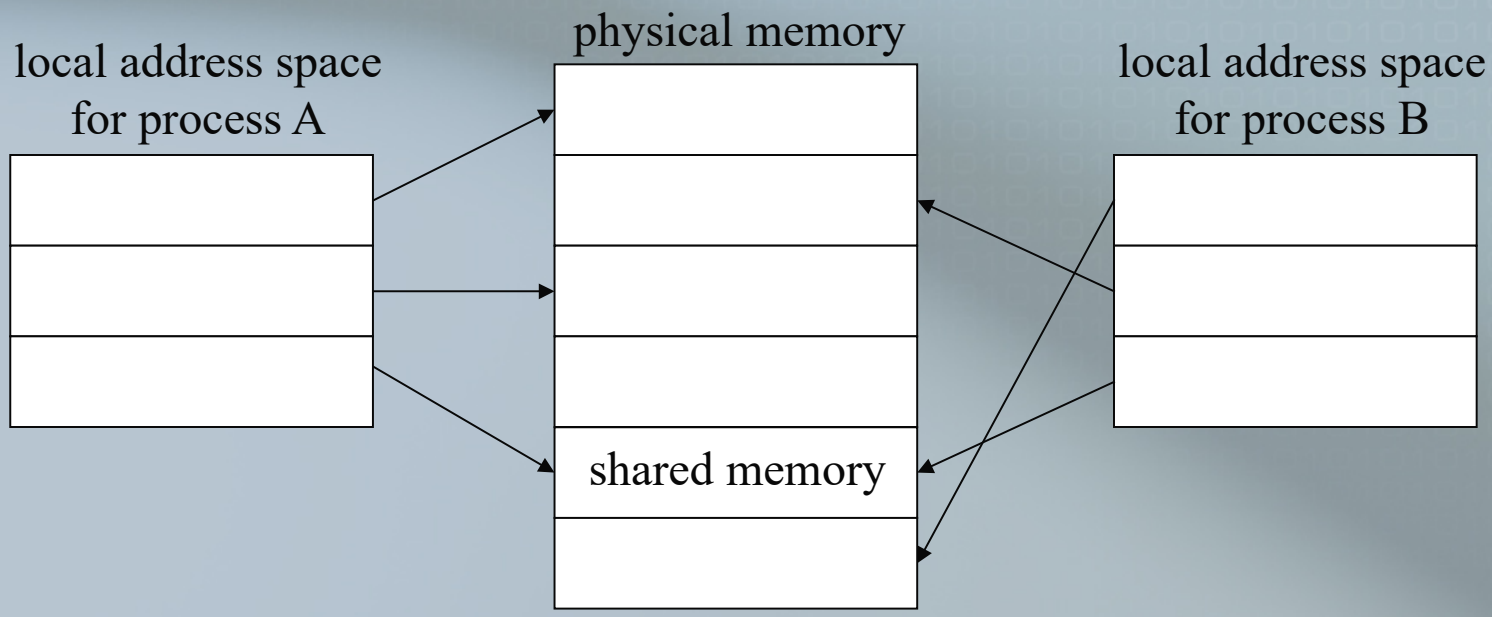
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch13]$ ./sem1 1 &
[1] 21161
[kevin@localhost ch13]$ ./sem1
process2 in     process2 out
process1 in     process1 out
process2 in     process2 out
process1 in     process1 out
process2 in     process2 out
process1 in     process1 out
process2 in     process2 out
process1 in     process1 out
process2 in     process2 out
process1 in
21162 - finished
[kevin@localhost ch13]$ process1 out

21161 - finished
```

Shared Memory Concept

- 共享記憶體是由 IPC 為一程序所建立的特殊記憶體位址，其他的程序可以將此相同的 shared memory 區段納入自己的位址空間中，所有的程序皆可存取這些記憶體位址，就像是由自己定址一樣。



Shared Memory Function (1/5)

- #include <sys/sem.h>
- #include <sys/type.h>
- #include <sys/ipc.h>

- int shmget(key_t *key*, size_t *size*, int *shmflg*);

  建立 shared memory 。
- void *shmat(int *shm_id*, const void **shm_addr*, int *shmflg*);

  允許程序對 shared memory 存取。
- int shmdt(const void **shm_addr*);

  讓目前的程序從 shared memory 脫離出來。
- int shmctl(int *shm_id*, int *cmd*, struct shmid_ds **buf*);

  用來改變 shared memory 。

Shared Memory Function (2/5)

- int shmget(key_t *key*, size_t *size*, int *shmflg*);
  - *key*: 用來為 shared memory 命名。
  - *size*: 需要的 shared memory 大小，以 byte 為單位。
  - *shmflg*: shared memory 的權限。在建立新的 shared memory 時要加上 IPC_CREATE 。

Shared Memory Function (3/5)

- void *shmat(int *shm_id*, const void **shm_addr*, int *shmflg*);
  - *shm_id*: Shared memory id.
  - *shm_addr*: Shared memory 加到目前程序中的位址，通常為一 null 指標。
  - *shmflg*: 一般設為 0 即可。

Shared Memory Function (4/5)

- int shmdt(const void **shm_addr*);
  - *shm_addr*: shmat 傳回的位址。

Shared Memory Function (5/5)

- int shmctl(int *shm_id*, int *cmd*, struct shmid_ds **buf*);
  - *shm_id*: Shared memory id.
  - *cmd*:IPC_RMID: 刪除 shared memory 區段。
  - struct shmid_ds {

    uid_t shm_perm.uid;

    uid_t shm_perm.gid;

    mode_t shm_perm.mode;

    }

Shared Memory Example (1/6)

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#include "shm_com.h"

int main()
{
    int running = 1;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;

    srand((unsigned int)getpid());

    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);

    if (shmid == -1) {
        fprintf(stderr, "shmget failed\n");
        exit(EXIT_FAILURE);
    }
```

shm_com.h

```c
#define TEXT_SZ 2048

struct shared_use_st {
    int written_by_you;
    char some_text[TEXT_SZ];
};
```

Shared Memory Example (2/6)

```
shared_memory = shmat(shmid, (void *)0, 0);
if (shared_memory == (void *)-1) {
    fprintf(stderr, "shmat failed\n");
    exit(EXIT_FAILURE);
}

printf("Memory attached at %X\n", (int)shared_memory);

shared_stuff = (struct shared_use_st *)shared_memory;
shared_stuff->written_by_you = 0;
while(running) {
    if (shared_stuff->written_by_you) {
        printf("You wrote: %s", shared_stuff->some_text);
        sleep( rand() % 4 ); /* make the other process wait for us ! */
        shared_stuff->written_by_you = 0;
        if (strncmp(shared_stuff->some_text, "end", 3) == 0) {
            running = 0;
        }
    }
}
```

Shared Memory Example (3/6)

```
    if (shmdt(shared_memory) == -1) {
        fprintf(stderr, "shmdt failed\n");
        exit(EXIT_FAILURE);
    }

    if (shmctl(shmid, IPC_RMID, 0) == -1) {
        fprintf(stderr, "shmctl(IPC_RMID) failed\n");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

Shared Memory Example (4/6)

```
int main()
{
    int running = 1;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    char buffer[BUFSIZ];
    int shmid;

    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);

    if (shmid == -1) {
        fprintf(stderr, "shmget failed\n");
        exit(EXIT_FAILURE);
    }

    shared_memory = shmat(shmid, (void *)0, 0);
    if (shared_memory == (void *)-1) {
        fprintf(stderr, "shmat failed\n");
        exit(EXIT_FAILURE);
    }

    printf("Memory attached at %X\n", (int)shared_memory);
```

Shared Memory Example (5/6)

```
printf("Memory attached at %X\n", (int)shared_memory);

shared_stuff = (struct shared_use_st *)shared_memory;
while(running) {
    while(shared_stuff->written_by_you == 1) {
        sleep(1);
        printf("waiting for client...\n");
    }
    printf("Enter some text: ");
    fgets(buffer, BUFSIZ, stdin);

    strncpy(shared_stuff->some_text, buffer, TEXT_SZ);
    shared_stuff->written_by_you = 1;

    if (strncmp(buffer, "end", 3) == 0) {
            running = 0;
    }
}
```

```
    if (shmdt(shared_memory) == -1) {
        fprintf(stderr, "shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

Shared Memory Example (6/6)

kevin@localhost:~/linux_program/ch13 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch13]$ ./shm1 &
[1] 21231
[kevin@localhost ch13]$ Memory attached at F7035000
./shm2
Memory attached at F706E000
Enter some text: hello
You wrote: hello
waiting for client...
Enter some text: hi
You wrote: hi
waiting for client...
waiting for client...
Enter some text: good
You wrote: good
waiting for client...
waiting for client...
waiting for client...
Enter some text: bye
You wrote: bye
waiting for client...
waiting for client...
Enter some text: end
You wrote: end
[kevin@localhost ch13]$
```

Message Queue Concept

- Message queue 就像 named pipe 一樣，但沒有開啟與關閉 pipe 的複雜性。不過使用 message queue 也有類似 named pipe 的問題，如 pipe 的阻礙模式。

Message Queue Function (1/5)

- #include <sys/sem.h>
- #include <sys/type.h>
- #include <sys/ipc.h>

- int msgget(key_t *key*, int *msgflg*);

  建立與存取 message queue 。
- int msgsnd(int *msqid*, const void **msg_ptr*, size_t *msg_sz*, int *msgflg*);

  加入 message 至 message queue 。
- int msgrcv(int *msqid*, void **msg_ptr*, size_t *msg_sz*, long int *msgtype*, int *msqflg*);

  從 message queue 擷取 message 。
- int msgctl(int *msqid*, int *cmd*, struct msqid_ds **buf*);

  用來控制 message queue 。

Message Queue Function (2/5)

- int msgget(key_t *key*, int *msgflg*);
  - *key*: 為 message queue 命名。
  - *msgflg*: message queue 的權限。在建立新的 message queue 時要加上 IPC_CREATE 。

Message Queue Function (3/5)

- int msgsnd(int *msqid*, const void **msg_ptr*, size_t *msg_sz*, int *msgflg*);
  - *msqid*: Message queue id.
  - *msg_ptr*: 指向被傳送訊息的指標，必需以為 long int 型態起始。

```
struct my_message {
  long int my_msg_type;  /* message type, must be > 0 */
  char some_text[MAX_TEXT];  /* message data */
};
```

  - *msg_sz*: *msg_ptr* 的大小，不包括 long int 型態。
  - *msgflg*: 控制 message queue 到達下限的反應動作。一般設為 0 即可。

Message Queue Function (4/5)

- int msgrcv(int *msqid*, void **msg_ptr*, size_t *msg_sz*, long int *msgtype*, int *msqflg*);
  - *msqid*: Message queue id.
  - *msg_ptr*: 指向接收訊息的指標，必需以為 long int 型態起始。
  - *msg_sz*: *msg_ptr* 的大小，不包括 long int 型態。
  - *msgtype*: 設定為 0 則從 queue 中取得第一個訊息；設定大於 0 則與 queue 中第一個訊息相同型態的訊息將全被取回；小於 0 則第一個與 *msgtype* 值相同或小於之訊息將被取回。
  - *msqflg*: 一般設為 0 。

Message Queue Function (5/5)

- int msgctl(int *msqid*, int *cmd*, struct msqid_ds **buf*);
  - *msqid*: Message queue id.
  - *cmd:*IPC_RMID: 刪除 message queue 。
  - struct msqid_ds {

    uid_t msg_perm.uid;

    uid_t msg_perm.gid;

    mode_t msg_perm.mode;

    }

HPDS

Message Queue Example (1/5)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>


struct my_msg_st {
    long int my_msg_type;
    char some_text[BUFSIZ];
};
```

```
int main()
{
    int running = 1;
    int msgid;
    struct my_msg_st some_data;
    long int msg_to_receive = 0;

    msgid = msgget((key_t)1234, 0666 | IPC_CREAT);

    if (msgid == -1) {
        fprintf(stderr, "msgget failed with error: %d\n", errno);
        exit(EXIT_FAILURE);
    }
```

Message Queue Example (2/5)

```
    while(running) {
        if (msgrcv(msgid, (void *)&some_data, BUFSIZ,
                    msg_to_receive, 0) == -1) {
            fprintf(stderr, "msgrcv failed with error: %d\n", errno);
            exit(EXIT_FAILURE);
        }
        printf("You wrote: %s", some_data.some_text);
        if (strncmp(some_data.some_text, "end", 3) == 0) {
            running = 0;
        }
    }

    if (msgctl(msgid, IPC_RMID, 0) == -1) {
        fprintf(stderr, "msgctl(IPC_RMID) failed\n");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

Message Queue Example (3/5)

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MAX_TEXT 512

struct my_msg_st {
    long int my_msg_type;
    char some_text[MAX_TEXT];
};
```

```c
int main()
{
    int running = 1;
    struct my_msg_st some_data;
    int msgid;
    char buffer[BUFSIZ];

    msgid = msgget((key_t)1234, 0666 | IPC_CREAT);

    if (msgid == -1) {
        fprintf(stderr, "msgget failed with error: %d\n", errno);
        exit(EXIT_FAILURE);
    }
```

Message Queue Example (4/5)

```
    while(running) {
        printf("Enter some text: ");
        fgets(buffer, BUFSIZ, stdin);
        some_data.my_msg_type = 1;
        strcpy(some_data.some_text, buffer);

        if (msgsnd(msgid, (void *)&some_data, MAX_TEXT, 0) == -1) {
            fprintf(stderr, "msgsnd failed\n");
            exit(EXIT_FAILURE);
        }
        if (strncmp(buffer, "end", 3) == 0) {
            running = 0;
        }
    }

    exit(EXIT_SUCCESS);
}
```

Message Queue Example (5/5)

kevin@localhost:~/linux_program/ch13 [80x24]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
[kevin@localhost ch13]$ ./msg2
Enter some text: hi
Enter some text: hello
Enter some text: good
Enter some text: end
[kevin@localhost ch13]$ ./msg1
You wrote: hi
You wrote: hello
You wrote: good
You wrote: end
[kevin@localhost ch13]$
```

Homework

- 請在 O.S 中的生產者與消費者問題加入同步機制。