Temas de Concurrencia POSIX

75.59 - Técnicas de Programación Concurrente I

Facultad de Ingeniería - Universidad de Buenos Aires

Resumen

Memoria Compartida (I)

Creación del objeto memoria compartida

Función *shm_open()*

```
int shm_open ( const char* name,int oflag,mode_t mode );
```

- Parámetros:
  - name: nombre del objeto de memoria, por ejemplo */nombre*
  - oflag: O_RDONLY, O_RDWR, O_CREAT, O_EXCL, O_TRUNC
  - mode: permisos
- Retorna:
  - El file descriptor de la región de memoria en caso de éxito
  - -1 en caso de error, seteando la variable externa *errno*
- Observación:
  - Linkear con opción *rt*: agregar parámetro *-lrt* al comando de compilación

Memoria Compartida (II)

Establecer el tamaño de la región de memoria compartida

Función *ftruncate()*

```
int ftruncate ( int fd,off_t length );
```

- Parámetros:
  - fd: file descriptor de la memoria compartida obtenido con *shm_open()*
  - length: tamaño deseado (en bytes)
- Retorna:
  - 0 en caso de éxito
  - -1 en caso de error, seteando la variable externa *errno*
- Observación:
  - La región de memoria compartida tuvo que haberse creado con permiso de escritura

Memoria Compartida (III)

Mapeo de la memoria compartida al espacio de direcciones del proceso: función *mmap()*

```
void* mmap ( void* addr,size_t length,int prot,int flags,int fd,off_t
offset );
```

- Parámetros:
  - addr: dirección a la cual se quiere mapear; *NULL* para que sea elegida por el SO
  - length: tamaño en bytes de mapeo de la memoria; suele coincidir con el tamaño dado en *ftruncate()*
  - prot: protección sobre los datos que están siendo mapeados; PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
  - flags: determina si las modificaciones en los datos mapeados son visibles para otros procesos; ejemplo: MAP_SHARED
  - fd: file descriptor de la memoria compartida obtenido con *shm_open()*
  - offset: byte a partir del cual se cuenta *length* (generalmente 0)

Memoria Compartida (IV)

- Retorna:
  - Puntero a la memoria en caso de éxito
  - *(void *) -1* en caso de error, seteando la variable externa *errno*

Memoria Compartida (V)

- Acceso a la memoria compartida
  - Puntero estándar de C/C++
- Destrucción de la memoria compartida
  - Función *close()* para cerrar el file descriptor
  - Función *munmap()* para eliminar el mapeo generado por *mmap()*

    ```
    int munmap ( void* addr,size_t len );
    ```
  - Función *shm_unlink()* para remover el objeto IPC generado por *shm_open()*

    ```
    int shm_unlink ( const char* name );
    ```

Semáforos (I)

- POSIX ofrece dos tipos de semáforos
  - Semáforos con nombre: permiten sincronizar procesos que no tienen relación entre sí
  - Semáforos sin nombre: se usan para sincronismo entre *threads* POSIX; no se pueden usar para sincronizar procesos
- Veremos solamente semáforos con nombre

Semáforos (II)

Apertura e inicialización del semáforo

Función *sem_open()*

```
sem_t* sem_open ( const char* name,int oflag,mode_t mode,unsigned int
value );
```

- Parámetros:
  - name: nombre del semáforo
  - oflag: flags; O_CREAT, O_EXCL
  - mode: permisos sobre el semáforo
  - value: valor de inicialización
- Retorna:
  - Dirección del semáforo en caso de éxito
  - SEM_FAILED en caso de error, seteando la variable externa *errno*
- Observación:
  - Linkear con opción *rt*: agregar parámetro *-lrt* al comando de compilación

Semáforos (III)

- Operaciones:
  - Función *sem_wait()*: decrementa el semáforo

    ```
    int sem_wait ( sem_t* sem );
    ```
  - Función *sem_post()*: incrementa el semáforo

    ```
    int sem_post ( sem_t* sem );
    ```
- Destrucción del semáforo:
  - Función *sem_close()*: cierra el semáforo

    ```
    int sem_close ( sem_t* sem );
    ```
  - Función *sem_unlink()*: elimina el objeto IPC

    ```
    int sem_unlink ( const char* name );
    ```

Colas de Mensajes: Apertura (I)

Función *mq_open()*

```
mqd_t mq_open ( const char *name,int oflag,mode_t mode,struct mq_attr
*attr );
```

- Parámetros:
  - name: nombre de la cola
  - oflag: opciones OR modo de apertura (ejemplo: O_RDWR |O_CREAT)
  - mode: permisos (ejemplo 0644)
  - attr: atributos de la cola
- Retorna:
  - El descriptor de la cola en caso de éxito
  - *(mqd_t) -1* en caso de error, seteando la variable externa *errno*
- Observación:
  - Linkear con opción *rt*: agregar parámetro *-lrt* al comando de compilación

Colas de Mensajes: Apertura (II)

- Atributos de la cola de mensajes

```
struct mq_attr {
    long mq_flags;    /* Flags: 0 u O_NONBLOCK */
    long mq_maxmsg;   /* Cantidad maxima de mensajes en
        la cola */
    long mq_msgsize; /* Tamanio maximo del mensaje (en
        bytes) */
    long mq_curmsgs; /* Cantidad actual de mensajes en
        cola */
};
```

- Funciones para leer y escribir los atributos de la cola:
  - *mq_getattr()*
  - *mq_setattr()*

Colas de Mensajes: Envío de mensajes

Función *mq_send()*

```
int mq_send ( mqd_t mqdes,const char *msg_ptr,size_t msg_len,unsigned
msg_prio );
```

- Parámetros:
  - mqdes: descriptor de la cola obtenido con *mq_open()*
  - msg_ptr: puntero al mensaje a enviar
  - msg_len: tamaño (en bytes) del mensaje a enviar
  - msg_prio: prioridad del mensaje (los mensajes se ordenan en la cola de mayor prioridad a menor prioridad)
- Retorna:
  - 0 en caso de éxito
  - -1 en caso de error, seteando la variable externa *errno*

Colas de Mensajes: Recepción de mensajes

Función *mq_receive()*: toma de la cola el mensaje más viejo con la prioridad más alta

```
ssize_t mq_receive ( mqd_t mqdes,char *msg_ptr,size_t msg_len,unsigned
*msg_prio );
```

- Parámetros:
  - mqdes: descriptor de la cola obtenido con *mq_open()*
  - msg_ptr: buffer para almacenar el mensaje recibido
  - msg_len: tamaño (en bytes) del buffer
  - msg_prio: buffer para guardar la prioridad del mensaje recibido (puede ser NULL)
- Retorna:
  - Tamaño en bytes del mensaje recibido en caso de éxito
  - -1 en caso de error, seteando la variable externa *errno*

Colas de Mensajes: Destrucción de la cola

- Función *mq_close()*: cierra la cola de mensajes

```
int mq_close ( mqd_t mqdes );
```

- Función *mq_unlink()*: elimina la cola de mensajes

```
int mq_unlink ( const char *name );
```

Bibliografía

- *Unix Network Programming, Interprocess Communications*, W. Richard Stevens, segunda edición
- Manuales del sistema operativo