LabSO

Message queues

Message queues

- **Le code di messagi (*message queues*) sono linked-lists all'interno del kernel identificate da una “chiave”.**
- **Si deve utilizzare una chiave (da condividire tra I processi interessati) per ottenere un identificativo da usare poi per tutte le operazioni successive**
- **Una coda deve essere innanzitutto generata: i “permessi” sono analoghi a quelli utilizzati per il file-system**
- **In ogni coda esistente si possono aggiungere o recuperare messaggi**
- **Ogni messaggio è caratterizzato da:**
  - un tipo (numero “long”)
  - una lunghezza non negativa
  - un insieme di dati (bytes) di lunghezza corretta
- **Tipicamente è “autosincrona”: quando si legge si attende la presenza di un messaggio se già non c'è e quando si scrive si attende vi sia spazio a disposizione, se già non c'è. Questi comportamenti possono essere “configurati”.**
- **In caso di errore in scrittura/lettura oltre a restituire “-1” le funzioni corrispondenti valorizzano anche la variabile globale *errno* per indicare il tipo di errore**

Message queues

- **si utilizzano fondamentalmente 5 syscall:**
  - `ftok()`: per generare la “chiave” univoca
  - `msgget()`: restituisce un identificatore in base alla chiave passata, eventualmente creando una nuova coda
  - `msgsnd()`: aggiunge messaggi a una coda
  - `msgrcv()`: recupera messaggi da una coda
  - `msgctl()`: effettua diverse operazioni, per esempio si può usare per eliminare un messaggio da una coda

Message queues

```
#include <sys/ipc.h>

key_t ftok(const char *path, int id);
```

- **restituisce una chiave basandosi sul *path* (cartella o file) nel file-system (che deve esistere ed essere “accessibile”) e sull'*id* numerico**
- **la chiave dovrebbe essere univoca e sempre la stessa per ogni coppia <*path*,*id*> in ogni istante sullo stesso sistema (nota: non è formalizzato il comportamento per uno stesso path che riferisce un percorso/file rimosso e poi ricreato!)**
- **un metodo d'uso – soprattutto per evitare “conflitti” - può essere di generare un *path* (ad esempio un file) temporaneo univoco, usarlo, eventualmente rimuoverlo e sfruttare *id* per rappresentare diverse “categorie” di code (una sorta di “indice)**

Message queues

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgget(key_t key, int msgflg);
```

- **restituisce l'identificativo di una coda basandosi sulla chiave (*key*) passata come argomento**
- **come secondo argomento si possono usare i flag:**
  - IPC_CREAT: crea una coda se non esiste già, altrimenti restituisce l'identificato di quella già esistente
  - IPC_EXCL: (usato insieme al precedente) crea una nuova coda se non esiste, altrimenti fallisce
  - 0xxx: un numero “ottale” di permessi analogo a quello che si può usare nel file-system

Message queues

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgsnd(int msqid, const void *msgp, size_t
msgsz, int msgflg);
```

- **aggiunge un messaggio alla coda identificata da *msqid*, puntata dal buffer *msgp*, di dimensione *msgsz***
- **come flag in *msgflg* si ha:**
  - IPC_NOWAIT: NON attende vi sia spazio (disabilita il comportamento “bloccante”)

Message queues

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);
```

- **recupare un messaggio dalla coda identificata da *msqid*, mettendo i dati nel buffer *msgp*, fino a una dimensione massima *msgsz*, di tipo *msgtyp***
- **se il messaggio ha dimensione maggiore di quella richiesta la chiamata fallisce a meno si specifichi il flag MSG_NOERROR**
- **come tipo in *msgtyp* si ha:**
  - 0 (zero): per leggere il primo messaggio disponibile in generale
  - x (intero positivo): per leggere il primo messaggio del tipo corrispondente, oppure di tipo diverso se si specifica il flag MSG_EXCEPT
  - x (intero negativo): per leggere il primo messaggio disponibile il cui tipo ha un indice minore o uguale al valore assoluto di x
- **come tipo in *msgflg* si ha:**
  - IPC_NOWAIT: NON attende vi sia un messaggio (disabilita il comportamento “bloccante”)
  - MSG_EXCEPT: se *msgtyp* è un intero positivo legge un messaggio di tipo diverso da quello specificato
  - MSG_NOERROR: tronca il messaggio letto anzichè uscire con errore se la sua lunghezza è superiore al massimo specificato

Message queues

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

- **effettua un'azione sulla coda identificata da *msqid*, come specificato da *cmd*, valorizzando la struttura *buf* con dati di output (ad esempio: tempo di ultima scrittura, tempo di ultima lettura, numero di messaggi nella coda, etc.)**
- **tra i comandi:**
  - IPC_STAT: recupera informazioni dal kernel
  - IPC_SET: imposta alcuni parametri
  - IPC_RMID: rimuove immediatamente la coda
  - IPC_INFO: recupera informazioni generali sui limiti delle code nel sistema (ad esempio le dimensioni massime possibili)
  - MSG_INFO: analoga a IPC_INFO ma con alcuni contenuti differenti
  - MSG_STAT: analoga a IPC_STAT ma con alcune differenze

Message queues

- (esempio writer/reader con Makefile)

Message queues

- **Esercizio: creare un “gestore” di messaggi tramite code che lancia comandi esterni (possibilmente con parametri) e salva l'output e l'error su file esterni e un “analista” che dato il nome di un comando recupei nell'ordine output e/o error.**
  - Suggerimenti:
    - utilizzare il “nome” del comando come base per l'identificativo delle code
    - passare nel messaggio il nome del file da dove recuperare i dati