Sincronização e Comunicação Baseada em Mensagens

Walter Fetter Lages

w.fetter@ieee.org

Universidade Federal do Rio Grande do Sul

Escola de Engenharia

Departamento de Engenharia Elétrica

ENG04008 Sistemas de Tempo Real

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Introdução

- Utiliza uma única construção para sincronização e comunicação
- Diferentes semânticas devido a diferenças em 3 aspectos
  - Modelo de sincronização
  - Método de identificação dos processos
  - Estrutura das mensagens

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Sincronização de Processos

- Em sistemas baseados em mensagem existe uma sincronização implícita, pois a mensagem não pode ser recebida antes de ser enviada
  - Em sistemas com variáveis compartilhadas o receptor pode ler a variável antes dela ser escrita
- Modelos de sincronização
  - Assíncrono (sem espera)
  - Síncrono (rendezvous)
  - Invocação remota (rendezvous estendido)

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Modelos de Sincronização

- Eventos assíncronos podem ser utilizados para gerar comunicação síncrona

P1

```
asyn_send(message)
wait(acknowledgment)
```

P2

```
wait(message)
asyn_send(acknowledgment)
```

- Comunicações síncronas podem ser utilizadas para gerar invocação remota

P1

```
syn_send(message)
wait(reply)
```

P2

```
wait(message)
computa resposta
syn_send(reply)
```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Comunicação Assíncrona

- Potencialmente buffers infinitos são necessários para armazenar mensagens não lidas
- Usualmente o envio de mensagens é programado para esperar um reconhecimento
  - Comunicação síncrona
- Mais comunicações são explícitas, tornando os programas mais complexos
- É mais difícil provar a correção do sistema

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Identificação dos Processos

- Direta
  - Send <mensagem> to <identificação do processo>
- Indireta
  - Send <mensagem> to <caixa postal>
- Simétrica
  - Send <mensagem> to <identificação do destino>
  - Wait <mensagem>from<identificação da origem>
- Assimétrica
  - Wait <mensagem>

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Identificação Indireta

- Mapeamento na entidade intermediária
  - Diversos-para-um
    - Cliente-servidor
  - Diversos-para-diversos
    - Diversos clientes e diversos servidores
  - Um-para-um
    - Um cliente e um servidor
  - Um-para-diversos
    - Broadcast

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Estrutura da Mensagem

- Idealmente qualquer tipo de dado, pré-definido ou definido pelo usuário, deveria poder ser transmitido
  - Problemas de implementação, especialmente quando a representação dos dados inclui ponteiros
- Por questões de implementação a maioria dos sistemas operacionais exige que os dados sejam "convertidos"para bytes antes da transmissão

Filas de Mensagens SV

- `int msgget(key_t key,int msgflg)`
  - Uma nova fila é criada se `key` = `IPC_PRIVATE` ou `key` é inexistente
  - `msgflg` = permissões de acesso
- `int msgctl(int msqid,int cmd,`
  `struct msqid_ds *buf)`
  - Realiza diversas operações na fila
    `IPC_STAT`: obtém a estrutura `msqid_ds`
    `IPC_SET`: seta a estrutura `msqid_ds`
    `IPC_RMID`: remove a fila do sistema

Filas de Mensagens SV

- `int msgsnd(int msqid,struct msgbuf *msgp, int msgsz,int msgflg)`
- ```
  struct msgbuf {
      long mtype;
      char mtext[msgsz];
  };
  ```
- `int msgrcv(int msqid,struct msgbuf *msgp, int msgsz,long msgtyp,int msgflg)`
  - `msgtyp` =0: primeira mensagem na fila, > 0: primeira mensagem de tipo `msgtype`,< 0: primeira mensagem de tipo $\leq$ abs(`msgtype`)
  - `msgflg` => `IPC_NOWAIT`
  ```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Mensagens POSIX

- Mensagens assíncronas indiretas
- Filas de mensagens
- Cada fila pode ter vários escritores e leitores
- Mensagens podem ter prioridades associadas
- Cada fila de mensagens possui atributos
- Tamanho máximo da fila
- Tamanho máximo das mensagens
- Flags
  - `MQ_BLOCK`
  - `MQ_NONBLOCK`
- Número de mensagens na fila

Criação e Destruição de Filas

- `mqd_t mq_open(char *mq_name, int oflags, mode_t permissions, struct mq_attr *mq_attr)`
  - `permissions`
    - `FOR_READ`
    - `FOR_WRITE`
- `int mq_close(mqd_t mq)`
  - A fila continua existindo
- `int mq_unlink(char *mq_name)`

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Envio e Recepção

- `size_t mq_receive(mqd_t mq, char *msg_buffer, size_t buflen, unsigned int *msgprio)`
- `int mq_send(mqd_t mq, const char *msg, size_t msglen, unsigned int msgprio)`
- `int mq_notify(mqd_t mq, const struct sigevent *notification`
  - `notification=NULL` cancela notificação existente

struct sigevent

```
typedef struct sigevent
{
    sigval_t sigev_value;
    int sigev_signo;
    int sigev_notify;
    union
    {
        int _pad[SIGEV_PAD_SIZE];
        struct
        {
            void (*_function)(sigval_t)
            void *_attribute;
        } _sigev_thread;
    } _sigev_un;
} sigevent_t;
```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Atributos

- `int mq_getattr(mqd_t mq,`
  `struct mq_attr *attrbuf)`
- `int mq_setattr(mqd_t mq,`
  `const struct mq_attr *new_attrs,`
  `struct mq_attr *old_attrs)`
- `struct mq_attr`
  `{`
  `long mq_maxmsg;`
  `long mq_msgsize;`
  `long mq_flags; /* ignorado */`
  `long mq_curmsgs;`
  `};`