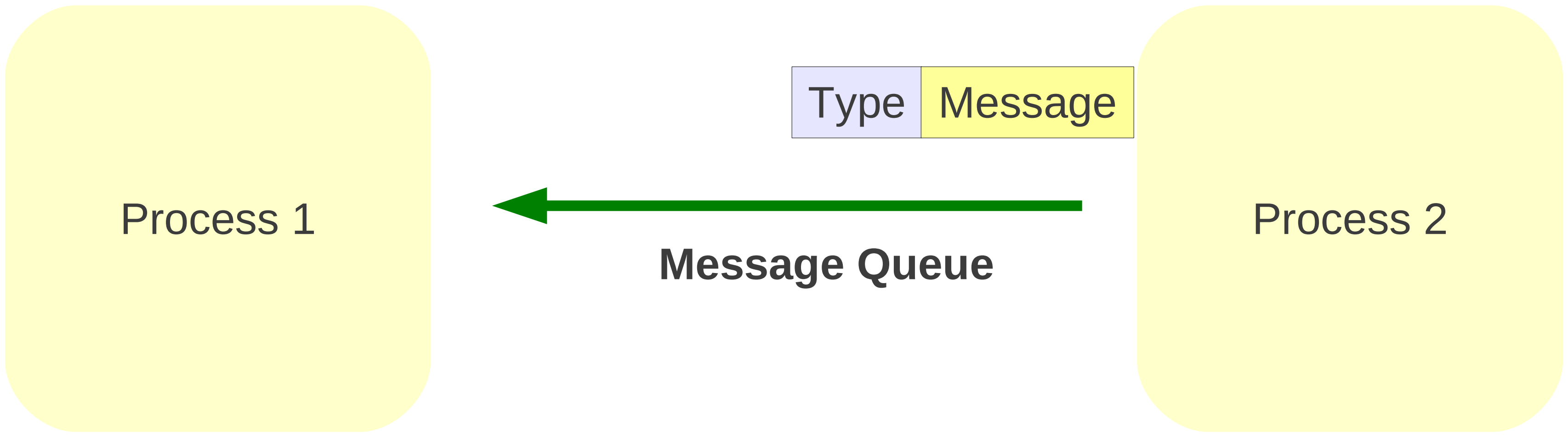
Message Queue (1A)

- Message Queue

Young Won Lim
12/13/2012

Message Queue



- send and receive messages
- queue messages for processing in an arbitrary order.
- When a message is sent, its text is copied to the message queue
- each IPC message
    - an explicit length (not like a pipe)
    - assigned a specific type

Message Queue System Call (1)

```
key_t ftok(); generate a key from a file name
int msgget(); connect to or create a queue

int msgsnd (); pass a message into a message queue
int msgrcv (); retrieve a message from a message queue

int msgctl(); to destroy a message queue

struct msgbuf {     // each message has 2 parts
    long mtype;     // positive long
    char mtext[1];  // any type
};
```

Message Queue System Call (2)

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

int msgget(key_t key, int msgflg); // returns msqid

int msgsnd
(int msqid, const void *msgp, size_t msgsz, int msgflg);

int msgrcv
(int msqid, void *msgp, size_t msgsz,long msgtyp, int msgflg);

int msgctl(int msqid, int cmd, struct msqid_ds *buf);

struct msgbuf {
    long mtype;
    char mtext[1];
};
```

Initialize the Message Queue (1)

```
int msgget(key_t key, int msgflg); // returns msqid
```

The msgget() function

- initializes a new message queue:
- return the message queue ID (msqid)
  of the queue corresponding to the key argument.

- key:
  - for a process to be able to identify the requested message queue
  - an arbitrary value or one that can be derived
    from a common seed at run time

- msgflg : octal permissions and control flags.

```
key=ftok("/home/bob/somefile", 'A')
```

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

ftok() converts a filename to a key value
that is unique within the system

Initialize the Message Queue (2)

```
int msgget(key_t key, int msgflg); // returns msqid
```

- If the key is IPC_PRIVATE, the call initializes a new instance of an IPC facility that is private to the creating process.

- IPC_CREAT - tries to create the message queue if it does not exist
- IPC_CREAT | IPC_EXCL flags - fails if the facility already exists
- Without IPC_CREAT or IPC_EXCL - return the existing queue ID
- Without IPC_CREAT and no existing queue - fails
- These can be combined with the octal permission modes

- `msqid = msgget(ftok("/tmp", 'A'), (IPC_CREAT | IPC_EXCL | 0400));`

Controlling Message Queues

int msgctl(int msqid, int cmd, struct msqid_ds *buf);

IPC_STAT (buf)
IPC_SET (buf)
IPC_RMID (buf)

The owner or creator can alter the permissions
and other characteristics of a message queue

cmd (argument → buf)

- IPC_STAT to get status of the queue
- IPC_SET to set the owner's user and group ID, the permissions,
  and the size (in number of bytes) of the message queue
- IPC_RMID to remove the message queue specified by the msqid

```
struct ipc_perm
{
  key_t  key;
  ushort uid;   /* owner euid and egid */
  ushort gid;
  ushort cuid;  /* creator euid and egid */
  ushort cgid;
  ushort mode;  /* access modes see mode flags below */
  ushort seq;   /* slot usage sequence number */
};
```

```
/* one msqid structure for each queue on the system */
struct msqid_ds {
    struct ipc_perm msg_perm;
    struct msg *msg_first;   /* first message on queue */
    struct msg *msg_last;    /* last message in queue */
    time_t msg_stime;        /* last msgsnd time */
    time_t msg_rtime;        /* last msgrcv time */
    time_t msg_ctime;        /* last change time */
    struct wait_queue *wwait;
    struct wait_queue *rwait;
    ushort msg_cbytes;
    ushort msg_qnum;
    ushort msg_qbytes;       /* max number of bytes on queue */
    ushort msg_lspid;        /* pid of last msgsnd */
    ushort msg_lrpid;        /* last receive pid */
};
```

Send & Receive Messages (1)

```
int msgsnd
(int msqid, const void *msgp, size_t msgsz, int msgflg);

int msgrcv
(int msqid, void *msgp, size_t msgsz,long msgtyp, int msgflg);
```

msgp
a pointer to a structure that contains
the type of the message and its text

Example :

```
 struct mymsg {
      long      mtype;    /* message type */
      char mtext[MSGSZ];  /* message text of length MSGSZ */
}
```

```
msgsz = sizeof(struct mymsg) - sizeof(long)
```

Send & Receive Messages (2)

```
int msgsnd
(int msqid, const void *msgp, size_t msgsz, int msgflg);

int msgrcv
(int msqid, void *msgp, size_t msgsz,long msgtyp, int msgflg);
```

msgtyp in msgrcv()

Zero	retrieve the next message on the queue, regardless of its mtype.
Positive	Get the next message with an mtype equal to the specified msgtyp.
Negative	Retrieve the first message on the queue whose mtype field is ≤ the absolute value of the msgtyp argument.

```
struct mymsg {
      long      mtype;     /* message type */
      char mtext[MSGSZ];  /* message text of length MSGSZ */
}
```

Reference

References

[1] http://en.wikipedia.org/
[2] http://beej.us/guide/bgipc/output/html/multipage/mq.html#mqwhere
[3] http://www.cs.cf.ac.uk/Dave/C/node25.html
[4] http://tldp.org/LDP/lpg/node21.html