Interprocess Communications

- System V interprocess communication (IPC) mechanism
  - Message Queues
  - Semaphores
  - Shared Memory
- Summary of processes related concepts

- Readings
  - APUE 15.6 —15.9

Why Other IPC Mechanisms

- Pipes/sockets
  - FIFO semantics
- Signals: sending flags
- Sometimes, we want something beyond FIFO
  - FIFO with tags (message queue)
  - File semantics: the content is always there unless it is modified explicitly (shared memory)
  - Once concurrency is allowed in shared data, we will need a way to protect (lock) the data (semaphore)

Message Queues

- What are they?
  - Similar to the FIFO pipes, except that a tag (type) is matched when reading/writing.
    - Allowing cutting in line (I am only interested in a particular type of message)
    - Equivalent to merging multiple FIFO pipes in one.

Message Queues

- Creating a message queue:
  - int msgget(key_t key, int msgflag);
  - Key can be any large number. But to avoiding using conflicting keys in different programs, use ftok() (the key master).
    - key_t ftok(const char *path, int id);
      - Path point to a file that the process can stat
      - Id: project ID, only the last 8 bits are used
  - Msgflag controls how message queue should be created
    - Access mode
    - What if a queue exists with the same key

Message Queues

- A linked list of messages stored within the kernel and identified by a message queue identifier.
  - Every message has a type field, and a nonnegative length, and the actual data bytes.
  - msgsnd puts a message at the end of the queue
  - msgrcv gets a message, may not follow FIFO order (can be based on type)
  - Has resource limits: MSGMAX, MSGMNB, MSGMNI, MSGTQL.

Message Queues

- Message queue operations

  int msgget(key_t, int flag)
  int msgctl(int msgid, int cmd, struct msgid_ds *buf)
  cmd: IPC_STAT, IPC_SET, IPC_RMID

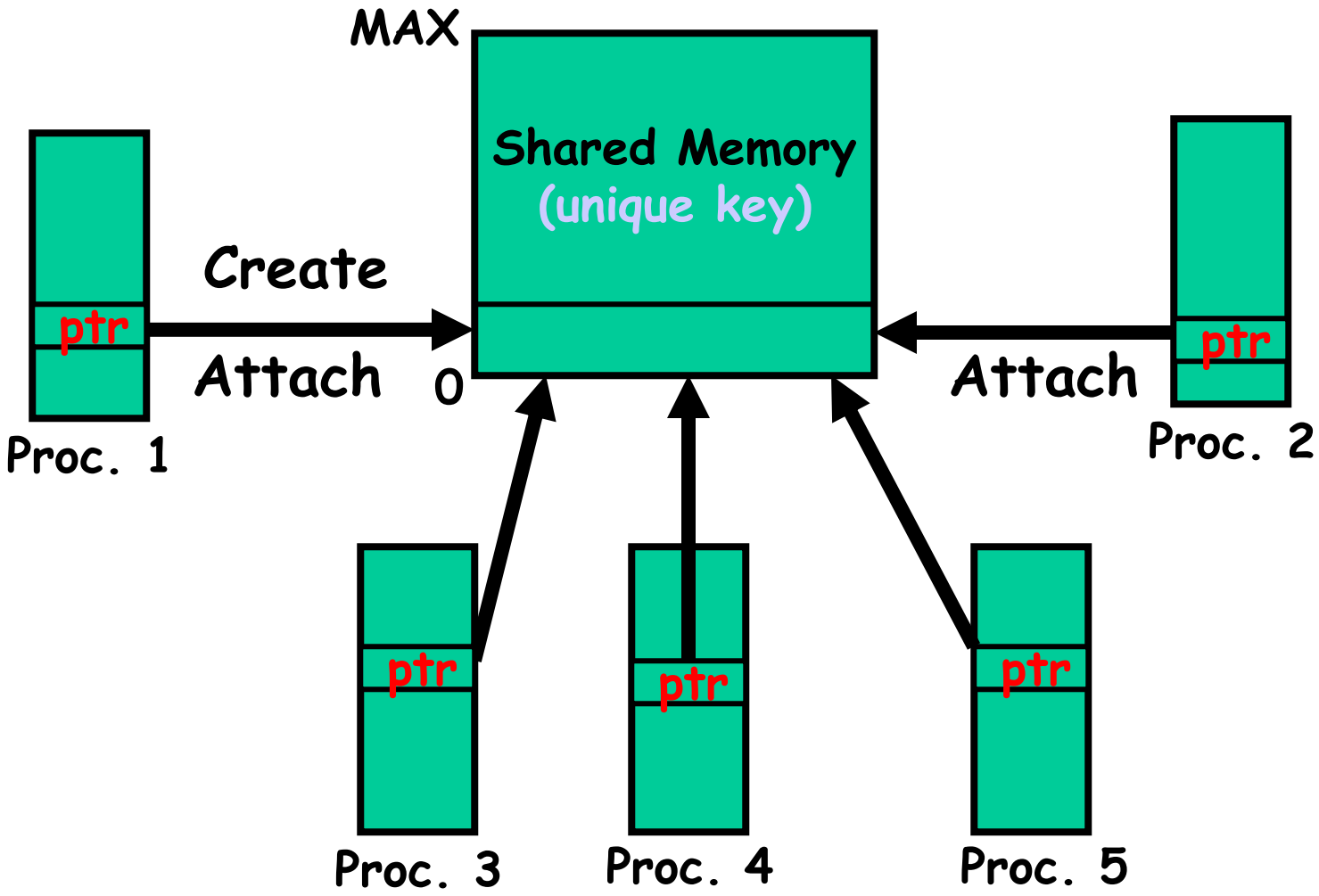
  int msgsnd(int msgid, const void *ptr, size nbytes, int flag);
  flag: IPC_NOWAIT

  int msgrcv(int msgid, void *ptr, size_t nbytes, long type, int flag);
  type = 0: first msg in queue
  type > 0: first msg whose msg type == type
  type < 0: first msg whose msg type is lowest value <= |type|

- Performance advantage is no longer there in newer systems (compared with pipe)

Shared Memory

Common chunk of read/write memory among processes



Creating Shared Memory

```
int shmget(key_t key, size_t size, int shmflg);
```

Example:

```
key_t key;
int shmid;

key = ftok("<somefile>", 'A');

shmid = shmget(key, 1024, 0644 | IPC_CREAT);
```

– See example shm_create.c

Attach and Detach Shared Memory

```
void *shmat(int shmid, void *shmaddr, int shmflg);
int shmdt(void *shmaddr);
```

Example:

```
key_t key;
int shmid;
char *data;

key = ftok("<somefile>", 'A');
shmid = shmget(key, 1024, 0644);
data = shmat(shmid, (void *)0, 0);
 …
shmdt(data);
```

– See example shm_attach.c

Deleting Shared Memory

```
int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

```
shmctl(shmid, IPC_RMID, NULL);
```

- See example shm_delete.c

Semaphores

- Managing concurrent access to shared memory segment.
- Using Semaphores
  - Creation: `semget( … )`
  - Incr/Decr/Test-and-set : `semop(…)`
  - Deletion: `semctl(semid, 0, IPC_RMID, 0);`

Online tutorial

http://beej.us/guide/bgipc/output/html/multipage/semaphores.html

Command-line IPC control

- ipcs
  - Lists all IPC objects owned by the user
- ipcrm
  - Removes specific IPC object

Summary of Process Related Concepts

Process Environment

- Command line arguments: argc, argv
- environ and getenv()
- getpid(), getuid(), getppid()
- How does a program access the second (command line) argument?
- How does a process access the variable you set in shell using commands such as “setenv TERM vt100”?
- How does a process know its parent’s process id? How does a parent know its children’s process ids?

Process Management

- fork, exit, wait, waitpid, execv
- How can a parent process know whether its child has executed successfully?
- How to determine whether execv runs a command successfully?

File Operations

- What are the related data structures for file operations?
- open, close, read, write, unlink, dup.
- How to redirect the standard input/output/error?

Interprocess Communications

- Pipe
- What kind of processes can communicate with pipes?
- How to implement “ps | grep duan | more”?
- Message queue
- Shared memory
- Semaphore

IPC

- Signal
- What is the typical default action for a signal?
- Blocking/unblocking a signal
  - sigset manipulation
    - sigfillset, sigemptyset, sigaddset, sigdelset, sigismember
  - the sigprocmask system call
- Install signal handler (can ignore a signal or use default handler)
  - signal
  - sigaction
- Sending signal: kill, alarm

Terminal I/O

- canonical mode
- noncanonical mode
  - tcgetattr, tcsetattr
  - termios data structure

Process Group, Session, and Controlling Terminal

- Related to job control
  - Who gets to access the keyboard? Who to send signal generated from the keyboard.
- Foreground and background processes
- Joining a group or creating a group: setpgid
- Making a group foreground or background
  - tcgetpgrp/tcsetpgrp