TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
EAST ASIA UNIVERSITY OF TECHNOLOGY

Chương 3

Tiến trình (Processes)

Nội dung:

- Khái niệm tiến trình
- Lập lịch tiến trình
- Các hoạt động trên tiến trình
- Các tiến trình hợp tác (Cooperating Processes)
- Giao tiếp liên tiến trình (Interprocess Communication)

3.1. Khái niệm tiến trình (process)

- Khi chương trình đang thực hiện
  + Được cung cấp tài nguyên (CPU, bộ nhớ, thiết bị vào/ra,. . .) để hoàn thành công việc
  + Tài nguyên được cấp khi:
    - Bắt đầu chương trình
    - Trong khi chương trình đang thực hiện
- Gọi là tiến trình (process) hay công việc (*job)* là tương tự nhau
- Tiến trình có thể chứa một hoặc nhiều tiểu trình
- Trách nhiệm của Hệ điều hành:
  + Đảm bảo hoạt động của tiến trình và tiểu trình (luồng)
  + Tạo/xóa tiến trình (người dùng, hệ thống)
  + Điều phối tiến trình
  + Cung cấp cơ chế đồng bộ, truyền thông và ngăn ngừa tình trạng bế tắc giữa các tiến trình

3.1. Khái niệm tiến trình (process)

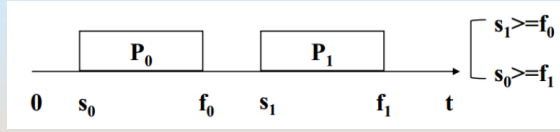
- Một HĐH thực hiện nhiều loại chương trình khác nhau:
  - Batch system: thực hiện các *job*
  - Time-shared system: thực hiện *user programs* hoặc *tasks*
- Một tiến trình (process) bao gồm:
  - program counter - bộ đếm chương trình
  - stack - ngăn xếp
  - data section - đoạn dữ liệu

3.1. Khái niệm tiến trình (cont...)

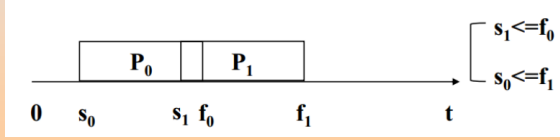
- **Tiến trình hệ thống**: được sinh ra khi thực hiện các lời gọi hệ thống
- **Tiến trình người sử dụng**: được sinh ra khi thực thi chương trình của người sử dụng
- **Tiến trình đơn luồng** : Là TT thực hiện *chỉ 1 luồng thực thi*
  ⇒ Cho phép thực hiện chỉ 1 nhiệm vụ tại 1 thời điểm
- **Tiến trình đa luồng** : Là TT có *nhiều luồng thực thi*
  ⇒ Cho phép thực hiện nhiều hơn 1 nhiệm vụ tại 1 thời điểm

3.1. Khái niệm tiến trình (cont...)

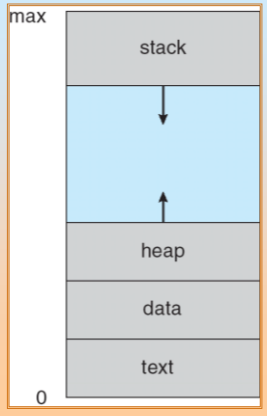
- **Tiến trình kế tiếp**



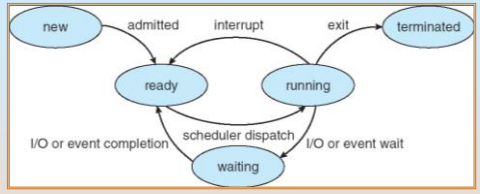
- **Tiến trình song song**



Tiến trình trong bộ nhớ

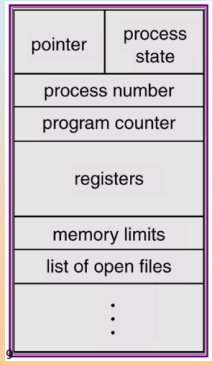


Các trạng thái tiến trình



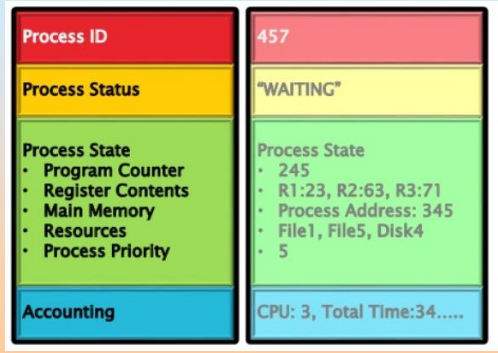
- Khi một tiến trình thực hiện, nó có thể thay đổi *trạng thái (state)*
  - **new**: Tiến trình đang được khởi tạo.
  - **running**: Tiến trình ở trong CPU. Các lệnh đang được thực hiện.
  - **waiting**: Tiến trình đang chờ sự kiện nào đó xuất hiện.
  - **ready**: Tiến trình đang chờ đến lượt được thực hiện bởi CPU.
  - **terminated**: Tiến trình kết thúc. Nó không biến mất cho đến khi một tiến trình khác đọc được trạng thái thoát của nó.

Khối điều khiển tiến trình
Process Control Block (PCB)



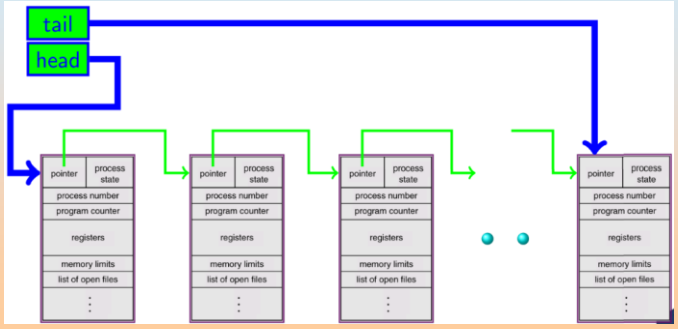
- Mỗi tiến trình được biểu diễn trong HĐH bởi một PCB. Mỗi PCB chứa các thông tin được gắn với mỗi tiến trình:
  - Trạng thái tiến trình
  - Bộ đếm chương trình
  - Các thanh ghi của CPU
  - Thông tin lịch trình CPU
  - Thông tin quản lý bộ nhớ
  - Thông tin sử dụng CPU, thời gian, các số hiệu tiến trình…
  - Thông tin trạng thái vào/ra
  - Con trỏ tới 1 PCB khác

Khối điều khiển tiến trình
Process Control Block (PCB)

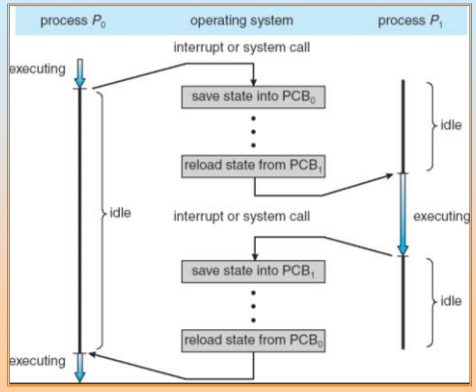


Khối điều khiển tiến trình
Process Control Block (PCB)

Danh sách tiến trình



CPU chuyển trạng thái giữa các tiến trình



CPU chuyển trạng thái giữa các tiến trình

- Khi CPU chuyển sang thực thi một process khác, hệ thống phải lưu trạng thái của process hiện tại đang thực thi và nạp trạng thái của process mới sẽ thực thi.
- Ngữ cảnh (context) của một process được biểu diễn trong khối PCB của process đó.
- Thời gian chuyển ngữ cảnh (context-switch time) là một trong các phí tổn (overhead) mà hệ thống phải gánh chịu; do đó, phải có chiến lược chuyển ngữ cảnh hợp lý để đạt hiệu quả xử lý cao.
- Chi phí chuyển ngữ cảnh phụ thuộc sự hỗ trợ cấp hardware, phụ thuộc phương thức quản lý bộ nhớ,...

CPU chuyển trạng thái giữa các tiến trình

- Lưu ngữ cảnh CPU, bao gồm thanh ghi lệnh - program counter (PC) và các thanh ghi khác.
- Cập nhật PCB của process đang thực thi: ghi nhận trạng thái hiện tại và một số thông tin cần thiết khác
- Chuyển PCB của process đang thực thi đến hàng đợi tương ứng: ready, waiting
- Thực hiện việc chọn process khác để thực thi (short-term scheduler)
- Cập nhật PCB của process được chọn thực thi.
- Phục hồi ngữ cảnh CPU của process được chọn (nếu có)

CPU chuyển trạng thái giữa các tiến trình

- Chuyển ngữ cảnh có thể xảy ra khi OS chiếm lại quyền điều khiển CPU, chẳng hạn như
  - System Call
    - » Được gọi tường minh trong chương trình (ví dụ: system call mở/đóng file). Process gọi system call có thể sẽ bị blocked chờ thực hiện system call.
  - Trap
    - » Một lỗi đã xảy ra. Process có thể chuyển vào trạng thái Exit và kết thúc thực thi.
  - Interrupt
    - » Quyền điều khiển chuyển sang cho Interrupt Handler.

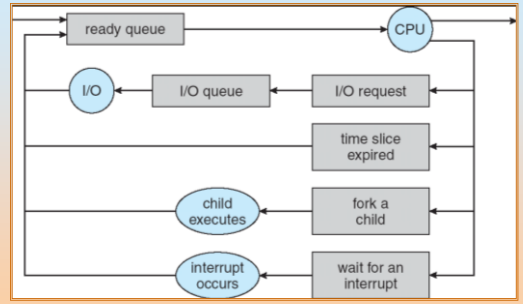
3.2. Lập lịch tiến trình (process scheduling)

- Mục tiêu của multiprogramming là có nhiều tiến trình cùng chạy tại mọi thời điểm để tối đa hóa sử dụng CPU.
- Mục tiêu của time-sharing là chuyển CPU giữa các tiến trình càng thường xuyên càng tốt để người sử dụng có thể tương tác với mỗi chương trình khi nó đang chạy.
- Một HĐH đơn processor chỉ có thể chạy 1 tiến trình.
- Nếu có nhiều tiến trình tồn tại, chúng phải đợi đến khi CPU rỗi và được lập lịch lại.

Các queue lập lịch tiến trình

- Job queue – tập hợp tất cả các tiến trình trong hệ thống.
- Ready queue – tập hợp tất cả các tiến trình cư trú trong bộ nhớ chính, đã sẵn sàng và chờ được thực hiện.
  - FIFO queue
  - Priority queue
  - Tree
  - Danh sách liên kết
- Device queues – tập hợp các tiến trình đang chờ một thiết bị vào/ra.
- Tiến trình có thể di trú giữa các queue khác nhau.

Sơ đồ lập lịch tiến trình

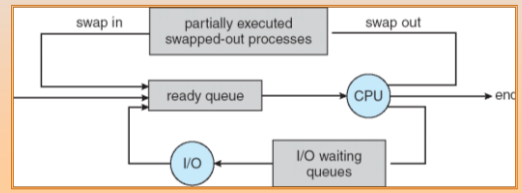


Các trình lập lịch - Schedulers

- Long-term scheduler (trình lập lịch dài kỳ)
  - còn được gọi là job scheduler.
  - lựa chọn những tiến trình nào nên được đưa từ đĩa vào trong ready queue.
  - được sử dụng đến rất không thường xuyên (seconds, minutes) ⇒ may be slow.
  - kiểm soát *mức đa chương trình* (*degree of multiprogramming*), vd: số tiến trình.
- Short-term scheduler (trình lập lịch ngắn kỳ)
  - còn được gọi là CPU scheduler.
  - lựa chọn tiến trình nào nên được thực hiện kế tiếp và phân phối CPU cho nó.
  - được sử dụng đến rất thường xuyên (milliseconds) ⇒ must be fast.

Các trình lập lịch (tiếp)

- Một số HĐH, như HĐH chia sẻ thời gian, có thể có thêm trình lập lịch trung kỳ (medium-term scheduler).
- Tư tưởng là thực hiện swapping (kỹ thuật hoán chuyển):
  - Đưa tiến trình ra khỏi bộ nhớ khi cần thiết
    - Khi cân đối giữa các tiến trình tính toán nhiều và tiến trình vào/ra nhiều
    - Khi cần giải phóng bộ nhớ cho việc khác
  - Sau đó đưa tiến trình trở lại bộ nhớ để thực hiện tiếp



Chuyển ngữ cảnh - Context Switch

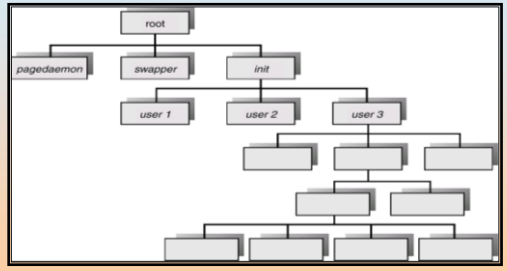
- Các tiến trình có thể được mô tả là:
  - *I/O-bound process* – sử dụng nhiều thời gian thực hiện vào/ra hơn việc tính toán, nhiều lần chiếm dụng CPU ngắn. Cần chuyển ngữ cảnh thường xuyên tại thời điểm bắt đầu và kết thúc I/O.
  - *CPU-bound process* – sử dụng nhiều thời gian cho việc tính toán hơn; ít lần chiếm dụng CPU dài. Cũng cần chuyển ngữ cảnh thường xuyên để tránh tr.hợp một tiến trình ngăn chặn các tiến trình khác sử dụng CPU.
- Khi CPU chuyển tới một tiến trình khác, hệ thống phải lưu trạng thái của tiến trình trước và nạp trạng thái đã lưu cho tiến trình mới.
- Thời gian chuyển ngữ cảnh phụ thuộc vào sự hỗ trợ phần cứng.

3.3. Các hoạt động trên tiến trình

- Các tiến trình trong hệ thống có thể thực hiện đồng thời, và chúng phải được tạo (create) và xóa (delete) một cách tự động.
- Do đó HĐH phải cung cấp kỹ thuật tạo và xóa tiến trình.

a) Sự tạo tiến trình - Process Creation

- Tiến trình cha (parent process) tạo ra các tiến trình con (children processes), chúng lần lượt tạo ra các tiến trình con khác tạo thành cây tiến trình (tree of processes).



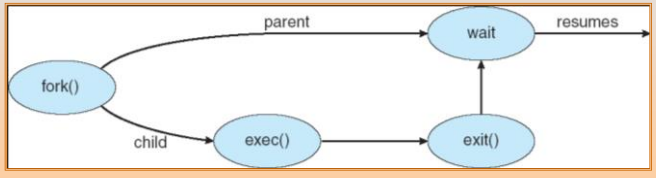
- Tạo tiến trình là một công việc "nặng nhọc" vì phải phân phối bộ nhớ và tài nguyên.

Sự tạo tiến trình (tiếp)

- Các lựa chọn chia sẻ tài nguyên (resource sharing)
  - Tiến trình cha và con chia sẻ tất cả tất cả các tài nguyên.
  - Tiến trình con được chia sẻ tập con các tài nguyên của tiến trình cha.
  - Tiến trình cha và con không có sự chia sẻ tài nguyên.
- Không gian địa chỉ (Address space)
  - Tiến trình con sao chép tiến trình cha.
  - Tiến trình con có một chương trình được nạp vào trong nó.
- Sự thực hiện (execution)
  - Tiến trình cha và con thực hiện đồng thời.
  - Tiến trình cha đợi cho đến khi tiến trình con kết thúc.
- UNIX examples

Các tiến trình trong UNIX

- **Fork()** - lệnh hệ thống tạo một tiến trình mới.
- **Exec()** - lệnh hệ thống được sử dụng sau lệnh fork để thay thế không gian bộ nhớ của tiến trình bởi một chương trình mới.



Khởi tạo tiến trình riêng biệt - C Program

```
int main()
{
pid_t  pid;
    /* fork another process */
    pid = fork();
    if (pid < 0) { /* error occurred */
            fprintf(stderr, "Fork Failed");
            exit(-1);
    }
    else if (pid == 0) { /* child process */
            execlp("/bin/ls", "ls", NULL);
    }
    else { /* parent process */
            /* parent will wait for the child to complete */
            wait (NULL);
            printf ("Child Complete");
            exit(0);
    }
}
```

VD: Lấy ID của tiến trình - C Program

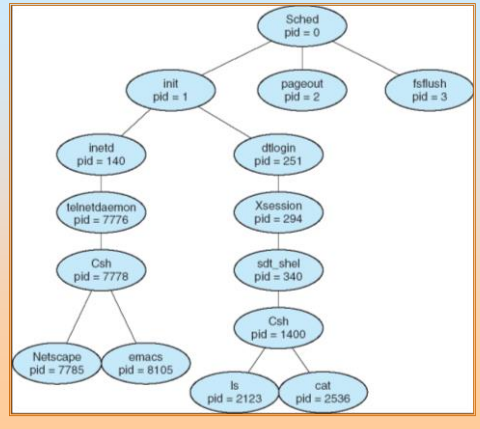
Chương trình để lấy id tiến trình và id tiến trình cha:

```
 #include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
    pid_t process_id;
    pid_t p_process_id;

    process_id = getpid();
    p_process_id = getppid();

    printf("The process id: %d\n",process_id);
    printf("The process id of parent function: %d\n",p_process_id);

    return 0;
}
```

Output

```
The process id: 3614
The process id of parent function: 3613
```

Ví dụ: Cây tiến trình trên HĐH Solaris



b) Sự kết thúc tiến trình

- Tiến trình thực hiện câu lệnh cuối cùng và yêu cầu HĐH tự kết thúc (**exit**).
  - Dữ liệu ra từ tiến trình con đến tiến trình cha (qua lệnh **wait**).
  - Các tài nguyên của tiến trình được HĐH phân phối lại.
- Tiến trình cha có thể chấm dứt việc thực hiện tiến trình con (**abort**).
  - Tiến trình con dùng quá tài nguyên được phân phối.
  - Nhiệm vụ mà tiến trình con thực hiện không còn cần thiết.
  - Tiến trình cha đang kết thúc. HĐH có thể lựa chọn:
    - Dừng tiến trình con. Xếp tầng sự chấm dứt (Cascading termination): tiến trình cháu cũng bị dừng,…
    - Tiến trình cháu tồn tại do được tiến trình ông chấp nhận.

3.4. Các tiến trình hợp tác

- Tiến trình *độc lập* (*Independent* process):
  - là tiến trình không thể ảnh hưởng hay bị ảnh hưởng bởi tiến trình khác thực thi trong hệ thống .
- Tiến trình *hợp tác* (*Cooperating* process):
  - có thể tác động hoặc chịu tác động bởi sự thực hiện của tiến trình khác.
  - vd: tiến trình này chia sẻ dữ liệu với tiến trình khác.

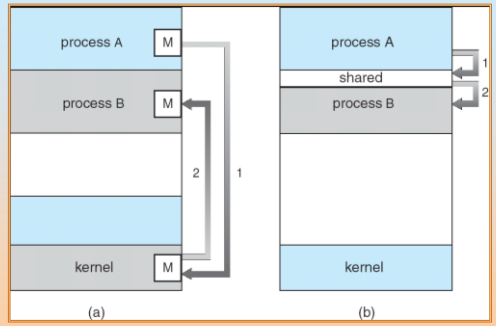
Các tiến trình hợp tác (tiếp)

- Các lợi điểm của tiến trình hợp tác
  - Chia sẻ thông tin - Information sharing
  - Tăng tốc độ tính toán - Computation speed-up
  - Mô-đun hóa - Modularity
  - Sự tiện lợi - Convenience (vd người sử dụng cùng thực hiện soạn thảo, in ấn, biên dịch song song)
- Mô hình các tiến trình hợp tác: tiến trình sản xuất (*producer* process) tạo ra các thông tin để tiến trình tiêu thụ (*consumer* process) sử dụng.
  - *unbounded-buffer* : giả thiết kích thước buffer vô hạn.
  - *bounded-buffer* : thừa nhận có một kích thước buffer cố định.

3.5. Giao tiếp liên tiến trình Interprocess Communication (IPC)

- Là cơ chế để các tiến trình giao tiếp và để đồng bộ các hành động của chúng mà không phải chia sẻ không gian địa chỉ chung.
- Khả năng IPC cung cấp 2 hoạt động:
  - **send** (*message*)– kích thước của message cố định hoặc biến đổi
  - **receive** (*message*)
- Nếu các tiến trình *P* và *Q* muốn giao tiếp, chúng cần phải:
  - thiết lập một *liên kết giao tiếp* (*communication link)* giữa chúng.
  - trao đổi các message qua các hoạt động send/receive.
- Sự thực hiện của communication link
  - physical (vd: shared memory, hardware bus)
  - logical (vd: logical properties)

Các mô hình giao tiếp



Giao tiếp trực tiếp

- Các tiến trình phải xác định rõ tên của nhau:
  - **send** (*P, message*) – gửi một message tới tiến trình P
  - **receive** (*Q, message*) – nhận message từ tiến trình Q
- Các đặc tính của communication link:
  - Các liên kết được thiết lập tự động.
  - Mỗi liên kết được gắn với duy nhất một cặp tiến trình giao tiếp với nhau.
  - Giữa mỗi cặp tiến trình tồn tại duy nhất 1 liên kết.
  - Liên kết thường là 2 chiều (bi-directional), hoặc có thể có 2 liên kết một chiều (unidirectional) P-to-Q, Q-to-P.

Giao tiếp gián tiếp

- Các message được gửi và nhận từ các mailbox (còn được gọi là port).
  - Mỗi mailbox có duy nhất một ***id***.
  - Các tiến trình chỉ có thể giao tiếp nếu chúng chia sẻ một mailbox.
- Các đặc tính của communication link:
  - Các liên kết được thiết lập chỉ khi các tiến trình chia sẻ một mailbox chung.
  - Một liên kết có thể được gắn với nhiều tiến trình.
  - Mỗi cặp tiến trình có thể chia sẻ một số communication link.
  - Liên kết có thể là 2 chiều hoặc 1 chiều.

Giao tiếp gián tiếp (tiếp)

- Các hoạt động:
  - tạo/xoá một mailbox
  - **send**(*A, message*) – gửi một message tới mailbox A
  - **receive**(*A, message*) – nhận một message từ mailbox A
- Vấn đề Mailbox sharing:
  - P_1, P_2, và P_3 chia sẻ mailbox A.
  - P_1, gửi; P_2 và P_3 nhận.
  - Tiến trình nào nhận được message?

Giao tiếp gián tiếp (tiếp)

- Giải pháp:
  - Một liên kết được gắn với tối đa 2 tiến trình.
  - Cho phép tại một thời điểm chỉ 1 tiến trình thực hiện nhận message.
  - Cho phép hệ thống tuỳ chọn tiến trình nhận. Tiến trình gửi được thông báo tiến trình nào nhận message.
  - Các tiến trình khác nhận được một bản copy.

Sự đồng bộ hóa - Synchronization

- Message passing có thể là có khóa (blocking) hoặc không khóa (non-blocking)
- **Blocking** được coi là *đồng bộ* (synchronous).
  - Blocking send: tiến trình gửi bị khóa đến khi message được nhận bởi tiến trình nhận hoặc bởi mailbox.
  - Blocking receive: tiến trình nhận/mailbox khóa đến khi nhận xong message.
- **Non-blocking** được coi là *không đồng bộ* (asynchronous).
  - Nonblocking send: tiến trình gửi gửi message rồi lại tiếp tục.
  - Nonblocking receive: tiến trình nhận/mailbox nhận được một message đúng hoặc vô dụng.

Buffering

- Hàng đợi của các message được gắn vào liên kết và được thực hiện bằng một trong ba cách:

1. Zero capacity: queue có độ dài lớn nhất bằng 0. Do đó, liên kết không thể có message đang chờ trong nó. Tiến trình gửi phải đợi tiến trình nhận.
2. Bounded capacity: queue có độ dài giới hạn bằng n. Khi 1 message được gửi, nó sẽ được đưa vào queue nếu queue chưa đầy, tiến trình gửi không phải đợi, và ngược lại.
3. Unbounded capacity: queue có độ dài vô hạn. Tiến trình gửi không bao giờ phải đợi.

VD: Windows 2000/XP

- Cung cấp sự hỗ trợ cho các môi trường đa xử lý (*subsystems*).
- Các chương trình ứng dụng có thể được coi là client của Windows 2000 subsystem server. Chúng giao tiếp sử dụng kỹ thuật message-passing.
- Khả năng message-passing trong Windows 2000 được gọi là LPC (local procedure call). LPC trong Windows 2000 cho phép giao tiếp giữa 2 tiến trình trên cùng máy.
- Mọi client gọi 1 subsystem cần có 1 kênh giao tiếp (connection port hay communication port) được cung cấp bởi port object.
- Sử dụng 3 kỹ thuật chuyển message qua cổng:
  - dùng message queue của cổng: message ≤ 256 byte
  - dùng bộ nhớ chia sẻ: dung lượng message lớn hơn
  - dùng kỹ thuật callback: khi client/server không thể đáp ứng lập tức

BÀI TẬP

1. Định nghĩa tiến trình và các thành phần của một tiến trình ?
2. Các bước HĐH khởi tạo một tiến trình ?
3. Nêu các trạng thái của tiến trình ?
4. Vẽ sơ đồ PCB ?
5. Vẽ sơ đồ chuyển ngữ cảnh giữa các tiến trình ?
6. Tại sao phải định thời tiến trình ?
7. Nêu các tác vụ của HĐH khi tạo mới một tiến trình ?
8. Nêu các tác vụ của HĐH khi kết thúc một tiến trình ?
9. Tại sao các tiến trình thường cộng tác với nhau ?
10. Khái niệm tiểu trình ?
11. Ưu điểm của tiểu trình ?
12. Sự khác biệt, mối quan hệ giữa tiến trình và tiểu trình ?