1. Một tiến trình chứa những thành phần gì?

Trả lời:

- Text section (program code)
- Data section (chứa global variables)
- Program counter, processor registers
- Heap section (chứa bộ nhớ cấp phát động)
- Stack section (chứa dữ liệu tạm thời)
  + Function parameters
  + Return address
  + Local variables

2. Tiến trình có những trạng thái nào? Cách tiến trình chuyển trạng thái?

Trả lời:

Trạng thái tiến trình:

- new: tiến trình vừa được tạo
- ready: tiến trình đã có đủ tài nguyên, chỉ còn cần CPU
- running: các lệnh của tiến trình đang được thực thi
- waiting: hay là blocked, tiến trình đợi I/O hoàn tất, tín hiệu
- terminated: tiến trình đã kết thúc

Cách tiến trình chuyển trạng thái :

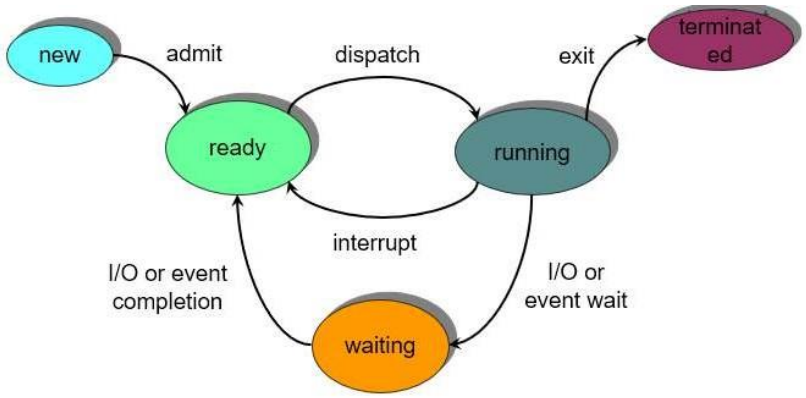
- Đầu tiên, khi vừa khởi tạo, tiến trình sẽ ở trạng thái là new.

- Thông qua bộ định thời dài hạn Long-term scheduling (hay còn gọi là bộ định thời công việc – Job Scheduler), tiến trình của chúng ta từ new sẽ được sắp xếp vị trí để "chui" vào trong hàng đợi ready.
- Ở một số hệ điều hành có thêm bộ định thời Medium-term Scheduler. Thông qua bộ định thời này, các tiến trình sẽ được swap-out (chuyển tiến trình từ bộ nhớ chính sang bộ nhớ phụ) và swap-in (chuyển tiến trình từ bộ nhớ phụ vào bộ nhớ). VD : Paging trong HĐH Windows, hoặc Swap trong Linux.
- Thông qua bộ định thời Short-term Scheduling (hay còn được gọi là Dispatcher), từ trạng thái ready tiến trình sẽ được sắp xếp để chuyển qua trạng thái running(là trạng thái chạy – hay trạng thái sử dụng CPU – của tiến trình).
- Trong khi đang ở trạng thái running, có 3 trạng thái tiếp theo mà tiến trình có thể đạt được tiếp theo :

  + waiting : Khi tiến trình đang chờ I/O (VD : Khi gọi hàm print(), scanf() trong C).

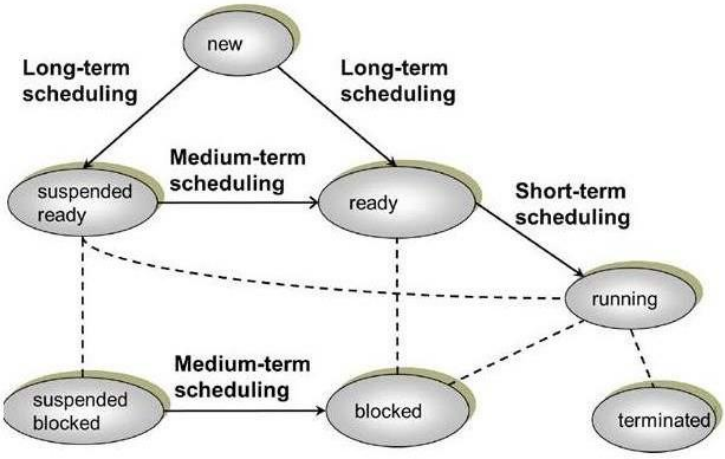
  + ready :Khi tiến trình bị interrupt (bị ngắt, không cho chạy nữa) bởi Short-term Scheduler. Các lý do ngắt có thể là : Ngắt thời gian (Clock Interrupt), Ngắt ngoại vi (I/O Interrupt), Lời gọi hệ thống (Operating System Call), Signal.

  + terminated : Khi ứng dụng thực thi xong : Khi gặp lệnh exit, khi thực thi lệnh cuối.
- Trong khi ở trạng thái waiting, tiến trình sẽ chuyển sang trạng thái ready(vào hàng đợi ready) sau khi đã thực thi xong I/O.
- Chuyển các trạng thái của tiến trình.



Chuyển đổi giữa các trạng thái của tiến trình

- Tiến trình chuyển trạng thái bằng các bộ định thời khác nhau.



- Có thể có >= 1 tiến trình ở trạng thái ready và waiting, tuy nhiên chỉ có duy nhất 1 tiến trình ở trạng thái runningtại một thời điểm nhất định mà ta đang xét).

3. Tại sao phải cộng tác giữa các tiến trình?

Trả lời:

Trong tiến trình thực thi, các tiến trình có thể cộng tác (cooperate) để hoàn thành các công việc.

Các tiến trình cộng tác với nhau để :

- Chia sẻ dữ liệu (information sharing).
- Tăng tốc độ tính toán (computational speedup).

  + Các mạng lưới máy tính sẽ hợp với nhau để tạo thành các cluster.

  + Nếu hệ thống có nhiều CPU, chia công việc tính toán thành nhiều công việc tính toán nhỏ chạy song song.
- Thực hiện một công việc chung.

+ Xây dựng một phần mềm phức tạp bằng cách chia thành các module/process hợp tác nhau.

4. PCB là gì? Dùng để làm gì?

Trả lời:

- PCB (Process Control Block) là một trong các cấu trúc dữ liệu quan trọng nhất của hệ điều hành. Mỗi tiến trình trong hệ thống đều được cấp phát một Process Control Block (PCB).
- PCB gồm:
    - Trạng thái tiến trình: new, ready, running,…
    - Bộ đếm chương trình
    - Các thanh ghi
    - Thông tin lập thời biểu CPU: độ ưu tiên, …
    - Thông tin quản lý bộ nhớ
    - Thông tin: lượng CPU, thời gian sử dụng,
    - Thông tin trạng thái I/O
- Tác dụng:
    - Hỗ trợ sự thực thi luân phiên giữa nhiều tiến trình
        - Hiệu suất sử dụng CPU
        - Thời gian đáp ứng
    - Phân phối tài nguyên hệ thống hợp lý
    - Tránh deadlock, trì hoãn vô hạn định
    - Cung cấp cơ chế giao tiếp và đồng bộ hoạt động các tiến trình
    - Cung cấp cơ chế hỗ trợ user tạo/kết thúc tiến trình

5. Tiểu trình là gì?

Trả lời:

- Tiểu trình là một đơn vị cơ bản sử dụng CPU gồm: Thread ID, PC, Registers, Stack và chia sẻ chung code, data, resourses (files)

6. Trình tự thực thi của tiến trình cha và tiến trình con?

Trả lời:

Giai đoạn 1: Tạo tiến trình con

- Tiến trình cha gọi hàm fork().
- Hệ điều hành sẽ tạo ra một bản sao của tiến trình cha.
- Bản sao này sẽ trở thành tiến trình con.
- Tiến trình con sẽ bắt đầu chạy từ vị trí mà hàm fork() được gọi.

Giai đoạn 2: Thực thi tiến trình con

- Tiến trình con sẽ thực thi các lệnh được viết trong mã nguồn của chương trình.
- Tiến trình con có thể tương tác với tiến trình cha thông qua các hàm hệ thống.
- Tiến trình con có thể tự kết thúc hoặc được tiến trình cha kết thúc.

Tương tác giữa tiến trình cha và tiến trình con

- Tiến trình cha và tiến trình con có thể tương tác với nhau thông qua các hàm hệ thống. Một số hàm hệ thống thường được sử dụng để tương tác giữa hai tiến trình bao gồm:

  + wait(): Tiến trình cha gọi hàm wait() để đợi tiến trình con kết thúc.

  + kill(): Tiến trình cha gọi hàm kill() để kết thúc tiến trình con.

  + pipe(): Tạo một ống dẫn giữa hai tiến trình để truyền dữ liệu.

  + shared memory(): Tạo bộ nhớ chia sẻ giữa hai tiến trình.

7. (Bài tập mẫu) Cho đoạn chương trình sau:

```
int main (int argc, char** argv)
{
```

```
        int i = 2;
        while (i < =5)
        {
                i++;
                if (i % 2 == 0)
                {
                        printf (“Hello”);
                        printf (“Hi”);
                }
                else
                {
                        printf (“Bye”);
                }
        }
        exit (0);
}
```

Hỏi trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái nào? Vẽ sơ đồ chuyển trạng thái trong quá trình thực thi?

Trả lời:

Trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái như sau: new – ready – running – waiting – ready – running – waiting – ready – running – waiting – ready – running – waiting – ready – running – terminated

8. Cho đoạn chương trình sau:

```
/* test.c */
int main(int argc, char** argv)
{
        int a;
        for (int i = 1; i < 5; i++)
        {
                if (i % 2 == 0)
                        printf(“Hello world\n");
                else a = 5*9;
        }
        exit(0);
}
```

Hỏi trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái nào? Vẽ sơ đồ chuyển trạng thái trong quá trình thực thi?

Trả lời:

Trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái như sau: New-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-terminated

9. Cho đoạn chương trình sau:

```
int main (int argc, char** argv)
{
        int i = 2;
        while (i < =5)
        {
                i++;
                if (i % 2 == 0)
                {
                        printf (“Hello”);
                        printf (“Hi”);
                }
                else
                {
                        printf (“Bye”);
                }
        }
        exit (0);
}
```

Hỏi trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái nào? Vẽ sơ đồ chuyển trạng thái trong quá trình thực thi?

Trả lời:

Trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trại thái như sau: new – ready – running – waiting – ready – running – waiting – ready – running – waiting – ready – running – waiting – ready – running – terminated

10. Cho đoạn chương trình sau:

```
int main (int argc, char** argv)
{
        int a, b, i;
        for (i = 16, i >=6; i --)
        {
                if (i % 3 == 0)
                {
                        printf (“Số %d chia hết cho 3”, i);
                }
                else
```

```
                {
                        a = b + i;
                }
        }
        exit (0);
}
```

Hỏi trong quá trình thực thi thì tiến trình khi chạy từ chương trình trên đã trải qua những trạng thái nào? Vẽ sơ đồ chuyển trạng thái trong quá trình thực thi?

Trả lời:

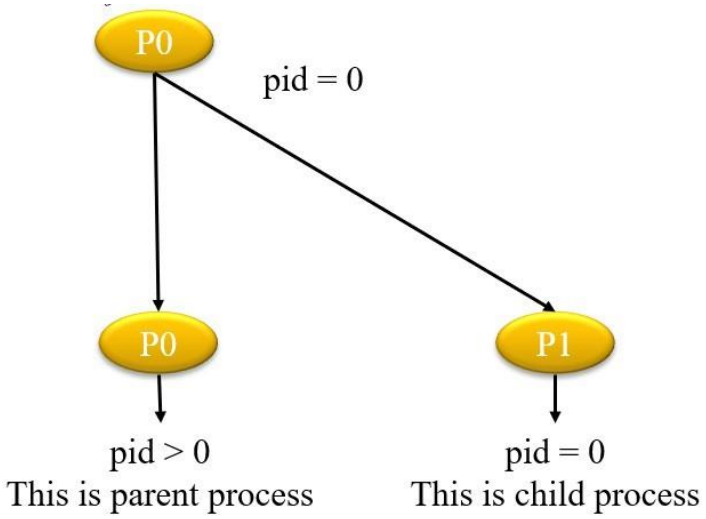
Trong quá trình thực thi thì tiến trình khi chạy từ chương trình nên đã trả qua những trạng thái như sau: new-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-running-waiting-ready-runing-waiting-ready-running-waiting-ready-runing-waiting-ready-runing-waiting-ready-terminated.

11. **(Bài tập mẫu) Cho đoạn code sau, hỏi khi chạy, bao nhiêu process được sinh ra và chương trình sẽ in ra những gì? Vẽ cây tiến trình khi thực thi đoạn chương trình sau**

```
#include <stdio.h>
#include <unistd.h>
int main (int argc, char *argv[]){
        int     pid;
        /* create a new process */
        pid = fork();
        if (pid > 0){
                printf("This is parent process");
                wait(NULL);
                exit(0);}
        else if (pid == 0)      {
                printf("This is child process");
                execlp("/bin/ls", "ls", NULL);
                exit(0);}
        else {   // pid < 0
                printf("Fork error\n");
                exit(-1);
        }
}
```

Trả lời:

Khi chạy đoạn chương trình trên, khi chạy hết sẽ có 2 process được sinh ra bao gồm 1 tiến trình cha và 1 tiền trình con. Theo chương trình trên thì tiến trình cha sẽ in ra dòng chữ "This is parent process"; và tiến trình con sẽ in ra dòng chữ "This is child process". Cây tiến trình khi thực thi đoạn chương trình trên như sau:

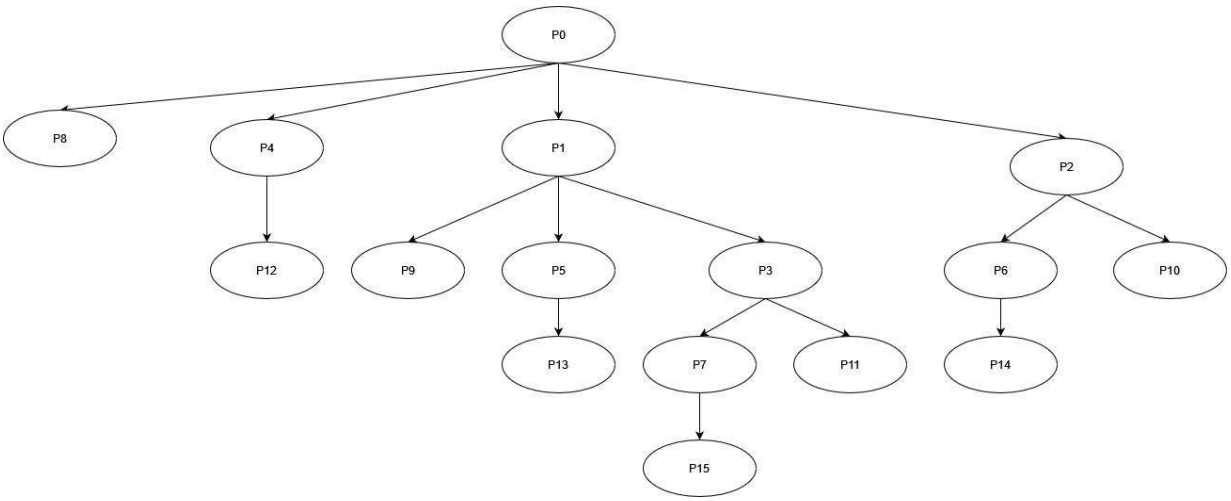


12. Cho đoạn code sau, hỏi khi chạy, bao nhiêu process (kể cả cha) được sinh ra? Vẽ cây tiến trình khi thực thi đoạn chương trình sau

```
int main()
{
        fork();
        fork();
        fork();
        fork();
        return 0;
}
```

Trả lời:

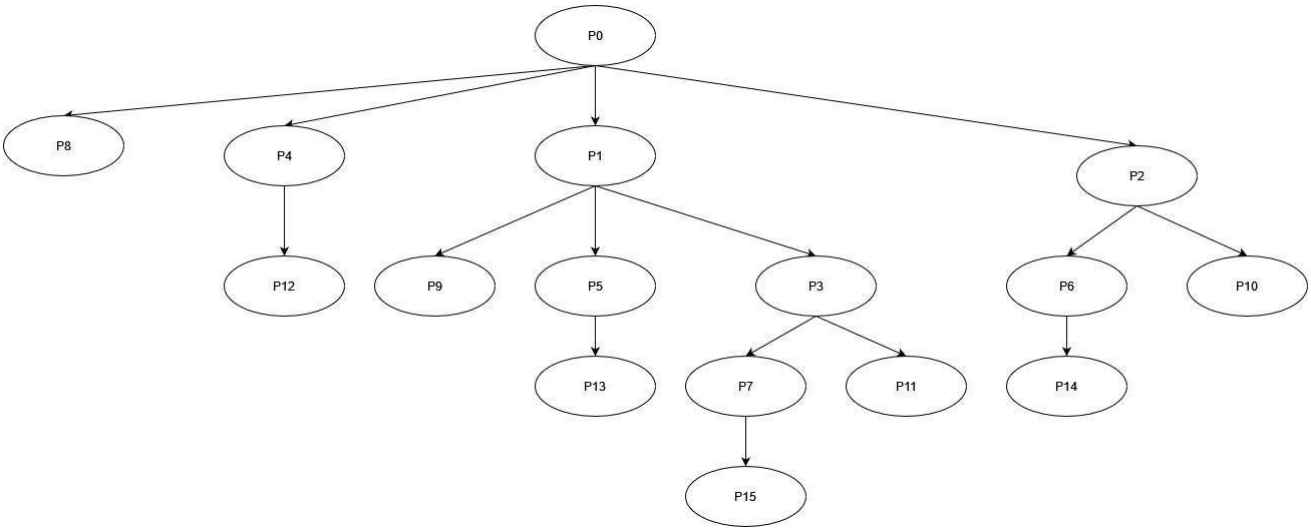
Có 16 process được in ra.



13. Cho đoạn code sau, hỏi khi chạy thì tiến trình được tạo ra từ chương trình trên sẽ in ra màn hình những gì? Vẽ cây tiến trình và những từ được in ra khi thực thi đoạn chương trình sau?

```
#include <stdio.h>
#include <unistd.h>
int main()
{
        int i;
        for (i = 0; i < 4; i++)
        {
                fork();
                printf("hello\n");
        }
        return 0;
}
```

Trả lời:



14. Cho đoạn code sau, hỏi khi chạy thì tiến trình được tạo ra từ chương trình trên sẽ in ra màn hình những gì? Vẽ cây tiến trình và những từ được in ra khi thực thi đoạn chương trình sau?

```
int main (int argc, char **argv)
{
        int pid;
        printf("Tiến trình cha \n");
        pid = fork();
        if (pid > 0)
        {
                fork();
                printf("Tiến trình cha \n");
        }
        else
        {
                printf("Tiến trình con \n");
                if(fork() > 0 )
                        printf("Tiến trình cha \n");
                else
                        printf("Tiến trình con \n");
        }
}
```

Trả lời:

Tiến trình cha

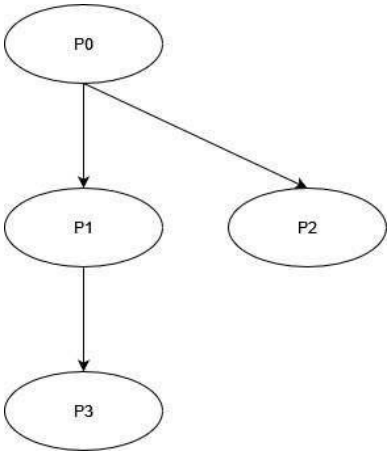
Tiến trình con

Tiến trình cha

Tiến trình cha

Tiến trình cha

Tiến trình con



15. Cho đoạn code chương trình sau:

```
if (fork() == 0)
{
   a = a + 5;
   printf("%d,%d\n", a, &a);
}
else
{
    a = a –5;
    printf("%d, %d\n", a, &a);
}
```

Giả sử u, v là các giá trị được in ra bởi process cha, và x, y là các giá trị được in ra bởi process con. Tìm mối quan hệ giữa u, v và x, y?

Trả lời:

Mối quan hệ giữa u, v và x, y là:

u = x + 5

v != y