MÔN HỌC: HỆ ĐIỀU HÀNH

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 3

1. Một tiến trình chứa những thành phần gì?

Trả lời:

Một tiến trình bao gồm:

- 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:

- 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 Waiting I/O hoàn tất, tín hiệu
- terminated: tiến trình đã terminated

Tiến trình trong máy tính chuyển trạng thái theo các cách sau:

- Từ **New** sang **Ready**: Khi tiến trình mới được tạo, nó sẽ chuyển sang trạng thái Ready.
- Từ **Ready** sang **Running**: Khi CPU cấp phát cho tiến trình, nó sẽ chuyển sang trạng thái Running.
- Từ **Running** sang **Waiting**: Khi tiến trình yêu cầu một tài nguyên nhưng chưa được đáp ứng vì tài nguyên chưa ready để cấp phát tại thời điểm đó; hoặc tiến trình phải chờ một sự kiện hay thao tác nhập/xuất.

- Từ **Waiting** sang **Ready**: Khi tài nguyên mà tiến trình yêu cầu trở nên ready để cấp phát; hay sự kiện hoặc thao tác nhập/xuất tiến trình đang Waiting hoàn tất.
- Từ **Running** sang **Terminated**: Khi tiến trình đã hoàn thành công việc của nó, nó sẽ chuyển sang trạng thái Terminated.

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

Trả lời:
Các tiến trình trong hệ thống máy tính cần phải cộng tác với nhau vì một số lý do sau đây:

- Chia sẻ thông tin: Nhiều tiến trình có thể cùng quan tâm đến những dữ liệu nào đó, do vậy hệ điều hành cần cung cấp một môi trường cho phép sự truy cập đồng thời đến các dữ liệu chung.
- Hợp tác hoàn thành tác vụ: Đôi khi để đạt được một sự xử lý nhanh chóng, người ta phân chia một tác vụ thành các công việc nhỏ có thể tiến hành song song. Thường thì các công việc nhỏ này cần hợp tác với nhau để cùng hoàn thành tác vụ ban đầu, ví dụ dữ liệu kết xuất của tiến trình này lại là dữ liệu nhập cho tiến trình khác.
- Tăng tốc tính toán (Computation Speedup): Các tiến trình cùng làm việc song song trên một hoặc nhiều máy để giải quyết bài toán chung.
- Đảm bảo tính đơn thể (Modularity): Chương trình được chia thành các đơn thể chức năng vận hành trong các tiến trình hoặc luồng khác nhau.

4. PCB là gì? Dùng để làm gì?

Trả lời:
PCB (Process Control Block) là một cấu trúc dữ liệu được sử dụng bởi hệ điều hành để lưu trữ tất cả thông tin về một tiến trình. Nó còn được gọi là Process Descriptor.
Khi một tiến trình được tạo ra (khởi tạo hoặc cài đặt), hệ điều hành tạo một PCB tương ứng. Thông tin trong PCB được cập nhật trong suốt quá trình dịch chuyển trạng thái của tiến trình.

Vai trò của PCB là trung tâm trong việc quản lý tiến trình: chúng được truy cập và/hoặc thay đổi bởi hầu hết các tiện ích, đặc biệt là các tiện ích liên quan đến việc lập lịch và quản lý tài nguyên. PCB lưu trữ dữ liệu cần thiết để cho việc quản lý tiến trình chính xác và hiệu quả. Mặc dù chi tiết của các cấu trúc này là tùy thuộc vào hệ thống nhưng các thành phần phổ biến thường rơi vào 3 loại chính: Process identification, Process state, Process control.
Trong suốt quá trình chuyển ngữ cảnh, tiến trình đang chạy sẽ bị dừng (stop) và tiến trình khác sẽ chạy. Kernel phải dừng việc thực thi của tiến trình đang chạy, copy giá trị trong thanh ghi vào PCB của tiến trình đó và cập nhật thanh ghi với giá trị từ PCB của tiến trình mới.

5. Tiểu trình là gì?

Trả lời:

Tiểu trình (thread) là một đơn vị xử lý cơ bản trong hệ thống. Mỗi tiểu trình xử lý tuần tự đoạn code của nó, sở hữu một con trỏ lệnh, tập các thanh ghi và một vùng nhớ stack riêng. Các tiểu trình chia sẻ CPU với nhau giống như cách chia sẻ giữa các tiến trình: một tiểu trình xử lý trong khi các tiểu trình khác chờ đến lượt.

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

Trả lời:

- tiến trình cha và con thực thi đồng thời (concurrently)
- tiến trình cha đợi đến khi các tiến trình con kết thúc

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:

Tiến trình khi chạy từ chương trình trên đã trải qua các trạng thái như sau: new – ready – running – waiting – ready – running – waiting – ready – running – 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 trạng thái như sau: new – 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 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

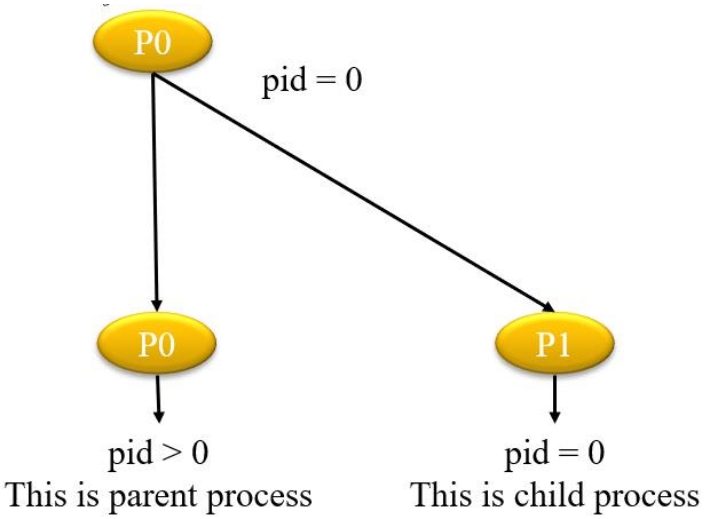
11. (Bài tập mẫu) Cho đoạn code sau, hỏi khi chạy, bao nhiêu process được sinh ra và chương trình sẽ in ra những gì? Vẽ cây tiến trình khi thực thi đoạn chương trình sau

```
#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:

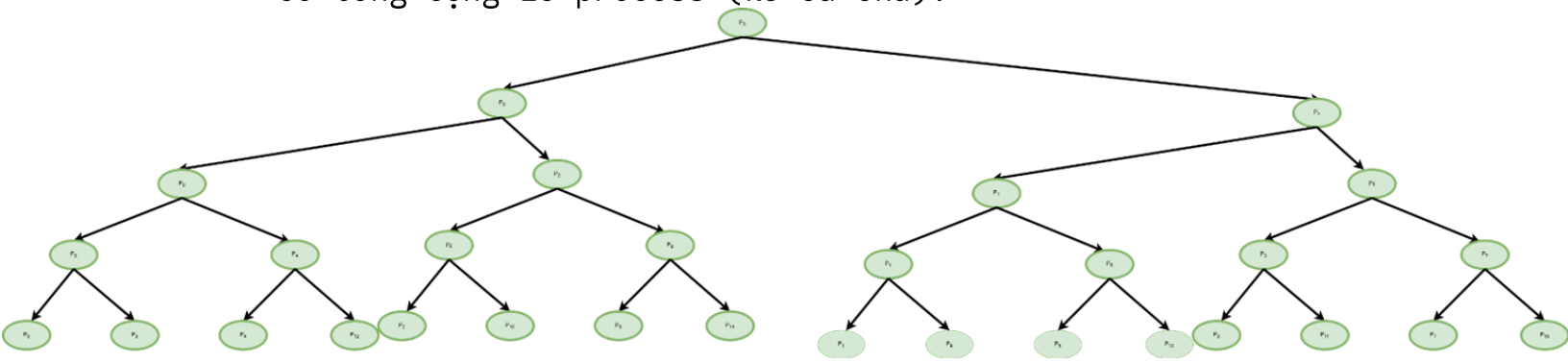


12. Cho đoạn code sau, hỏi khi chạy, bao nhiêu process (kể cả cha) được sinh ra? Vẽ cây tiến trình khi thực thi đoạn chương trình sau

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

Trả lời:

- Có tổng cộng 16 process (kể cả cha).



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:

Chương trình in ra màn hình:

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

hello

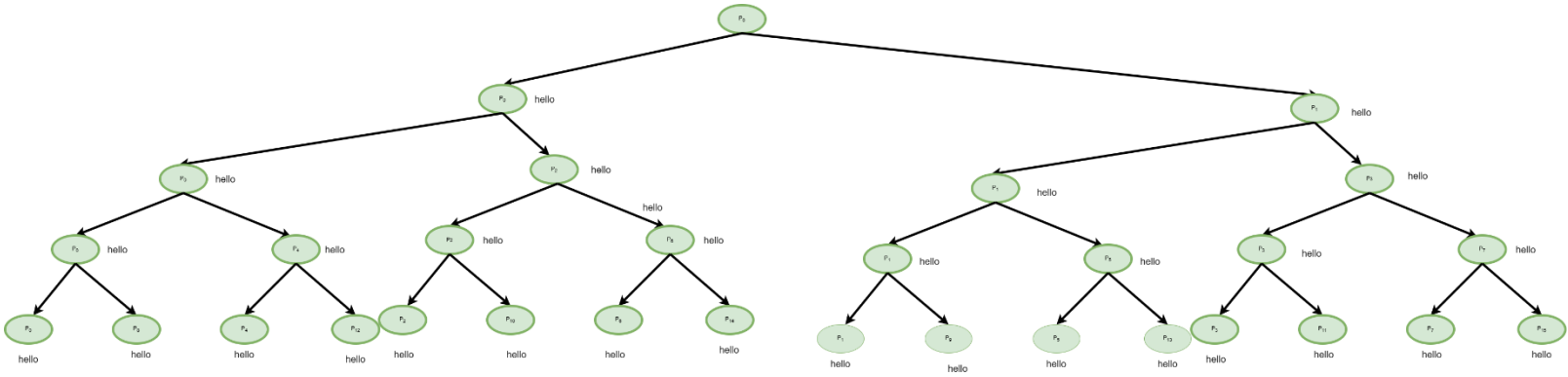
hello

hello

hello

hello

hello



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:

Chương trình in ra màn hình:

Tiến trình cha

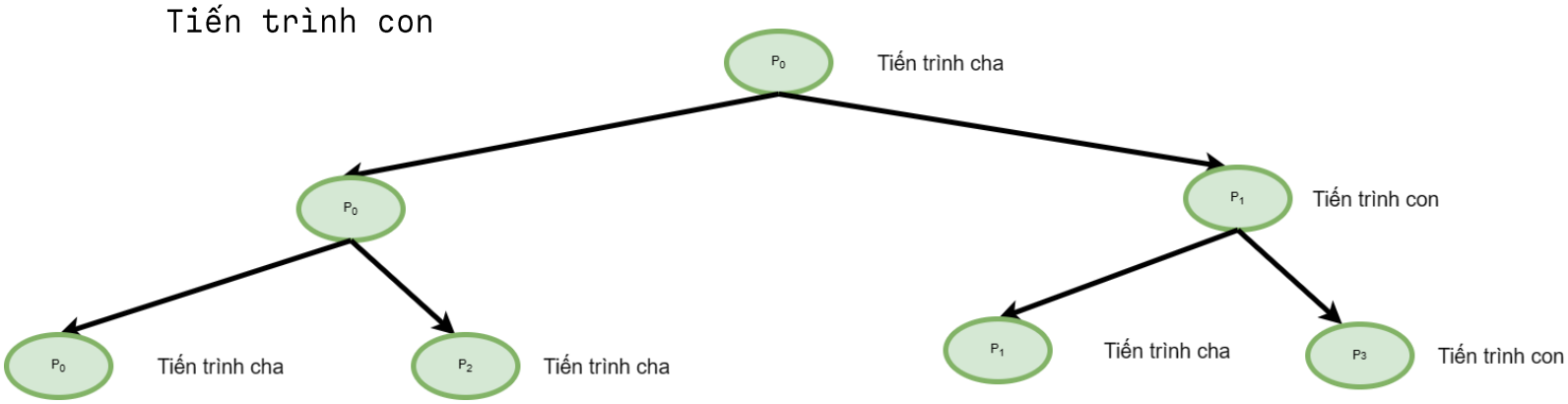
Tiến trình con

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:

- u và x luôn có giá trị khác nhau vì u và x là hai giá trị có công thức khác nhau của a.
- v và y luôn khác nhau do mỗi tiến trình có không gian địa chỉ riêng.