**­­ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

Text

Description automatically generated with medium confidence

**Nguyễn Lê Quỳnh Hương – 21520255**

IT007.N21.HTCL

**BÀI TẬP:**

**CHƯƠNG 3**

**Tiến trình**

**Giảng viên : Lê Hoài Nghĩa**

**Hồ Chí Minh, 2023**

**Câu 1 : Nêu cụ thể các trạng thái của tiến trình?**

/\* test.c \*/

int main(int argc, char\*\* argv)

{

printf(“Hello world\n");

scanf(“ Nhập c = %d”,&c);

exit(0);

}

* Chuỗi trạng thái của tiến trình test như sau (trường hợp tốt nhất):
  + new
  + ready
  + running
  + waiting (do chờ I/O khi gọi printf)
  + ready
  + running
  + waiting ( do chờ I/O scanf )
  + ready
  + running
  + terminated

**­**

**Câu 2 : Chương trình in ra những chữ gì ?**

#include <stdio.h>

#include <unistd.h>

int main (int argc, char \*argv[])

{

int pid;

pid = fork();

printf(“ so 1”);

printf(“ so 2”);

fork();

if (pid < 0){

printf(“hello”);

fork();

}else

fork();

printf(“bye”);

}

* Sử dụng hàm fork - > tạo ra các tiến trình con, hàm fork tạo ra một bản sao của tiến trình hiện tại và chạy đồng thời với tiến trình cha.
* Khi chương trình được chạy sẽ in ra "so 1" và "so 2" một lần.
* Hàm fork được gọi và tạo ra một tiến trình con. Cả tiến trình cha và tiến trình con đều sẽ in ra "bye" một lần. ( 2 chữ bye )
* Hai tiến trình sẽ gọi hàm fork thứ hai và tạo ra hai tiến trình con - > có tổng cộng 4 tiến trình đang chạy và mỗi tiến trình sẽ in ra "bye" một lần. ( 4 chữ bye )
* Bốn tiến trình sẽ gọi hàm fork cuối cùng và tạo ra thêm bốn tiến trình con -> có tổng cộng 8 tiến trình đang chạy và mỗi tiến trình sẽ in ra "bye" một lần nữa. ( 8 chữ bye )
* Chương trình này sẽ in ra "so 1" và "so 2" một lần, và in ra "bye" 14 lần, thứ tự xuất hiện của các chuỗi có thể khác nhau tùy thuộc vào việc hệ điều hành lên lịch chạy các tiến trình như thế nào.

**Câu 3 : Process control block chứa những thông tin gì?**

* 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

**Câu 4 : Các tác vụ đối với tiến trình?**

* Tạo tiến trình mới
  + Một tiến trình có thể tạo nhiều tiến trình mới thông qua một lời gọi hệ thống create-process
  + Tiến trình được tạo ra là tiến trình con
  + Tiến trình con nhận tài nguyên: từ HĐH hoặc tiến trình cha
  + Chia sẻ tài nguyên của tiến trình cha
    - Trình tự thực thi
    - Tiến trình cha và con thực thi đồng thời
    - Tiến trình cha đợi đến khi các tiến trình con kết thúc
* Kết thúc tiến trình
  + Tiến trình tự kết thúc
  + Tiến trình kết thúc do tiến trình khác
  + Hệ điều hành thu hồi tất cả các tài nguyên của tiến trình kết thúc

**Câu 5 : Tại sao phải định thời, có mấy loại bộ định thời?**

* + - * Định thời trong là một cơ chế quản lý tài nguyên máy tính, đảm bảo các tiến trình và ứng dụng được phân bổ thời gian xử lý công bằng và hiệu quả, giúp hệ thống hoạt động trơn tru và đáp ứng nhu cầu của người dùng nhanh chóng.
* Đa chương
  + Có vài tiến trình chạy tại các thời điểm
  + Mục tiêu: tận dụng tối đa CPU
* Chia thời
  + User tương tác với mỗi chương trình đang thực thi
  + Mục tiêu: tối thiểu thời gian đáp ứng
    - * Bộ đình thời
* Bộ định thời công việc (Job scheduler) hay bộ định thời dài (long-term scheduler)
* Bộ định thời CPU hay bộ định thời ngắn
* Bộ định thời trung gian
  + Đôi khi hệ điều hành (như time-sharing system) có thêm medium-term scheduling để điều chỉnh mức độ đa chương của hệ thống
  + Medium-term scheduler

**Câu 6 : Tiến trình có những trạng thái nào? Cách tiến trình chuyển trạng thái?**

* New: process vừa được tạo
* Ready: process đã có đủ tài nguyên, chỉ còn cần CPU
* Running: Các lệnh của tiến trình đang được thực thi
* Waiting (blocked): tiến trình đợi I/O hoàn tất, tín hiệu
* Terminated: tiến trình kết thúc

**Câu 7 : Tại sao phải cộng tác giữa các tiến trình?**

* Information sharing
* Computational speedup
* Thực hiện một công việc chung

**Câu 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?**

New - > ready - > running - > ready - > running -> waiting - > ready -> running - > ready - > running - > waiting - > terminated.

**Câu 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);

}

new - > ready - > running - > waiting -> ready -> running -> waiting - > ready - > running -> waiting -> ready -> running - > waiting -> terminated.

**Câu 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);

}

new -> ready -> running -> ready -> running - > waiting -> ready -> running -> ready -> running -> ready -> running -> waiting -> ready -> running -> ready -> running -> ready -> running -> waiting -> ready -> running -> ready -> running -> ready -> running -> waiting

**Câu 11: Cho đoạn code sau, hỏi khi chạy, bao nhiêu process (kể cả cha) được sinh ra?**

int main()

{

fork();

fork();

fork();

fork();

return 0;

}

* Tạo ra tổng cộng 16 tiến trình bao gồm cả tiến trình cha.
* Mỗi lần gọi fork() sẽ tạo ra một tiến trình mới, nhân đôi số lượng tiến trình.
* Sau lần gọi đầu tiên đến fork(), có 2 tiến trình.
* Sau lần gọi thứ hai, có 4 tiến trình.
* Sau lần gọi thứ ba, có 8 tiến trình.
* Sau lần gọi thứ tư, có 16 tiến trình.

**Câu 12 : Cho đoạn code sau, hỏi khi chạy thì tiến trình được tạo ra từ chương trình trên sẽ in ra màn hình những gì**

#include <stdio.h>

#include <unistd.h>

int main()

{

int i;

for (i = 0; i < 4; i++)

{

fork();

printf("hello\n");

}

return 0;

}

* In ra tổng cộng 30 dòng chữ "hello" trên màn hình.
* Mỗi lần gọi fork() sẽ tạo ra một tiến trình mới và nhân đôi số lượng tiến trình.
* Sau lần gọi đầu tiên đến fork(), có 2 tiến trình và in ra 2 dòng "hello".
* Sau lần gọi thứ hai, có 4 tiến trình và in ra thêm 4 dòng "hello".
* Sau lần gọi thứ ba, có 8 tiến trình và in ra thêm 8 dòng "hello".
* Sau lần gọi thứ tư, có 16 tiến trình, in ra thêm 16 dòng "hello".
* Tổng cộng sẽ có 30 dòng "hello" được in ra màn hình.

**Câu 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**

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");

}

}

* Tạo ra tổng cộng 4 tiến trình và in ra màn hình các dòng chữ sau:

“ Tiến trình cha

Tiến trình con

Tiến trình cha

Tiến trình cha

Tiến trình con

Tiến trình cha”

**Câu 14 : 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);

}

* fork() được gọi để tạo ra một tiến trình con.
* Nếu fork() trả về 0, - > tiến trình con và giá trị của a tăng lên 5 đơn vị.
* Nếu không phải là tiến trình cha, thì giá trị của a sẽ giảm đi 5 đơn vị.
* Nếu u và v là các giá trị được in ra bởi tiến trình cha và x và y là các giá trị được in ra bởi tiến trình con, thì mối quan hệ giữa chúng sẽ là: u = x - 10 và v = y.
* Giá trị của a trong tiến trình cha sẽ bằng giá trị của a trong tiến trình con trừ đi 10 đơn vị, còn địa chỉ của a trong cả hai tiến trình bằng nhau.