TÀI LIỆU HỆ ĐIỀU HÀNH **BUỔI 3: Những vấn đề về lý thuyết chương 3**

DÀNH CHO KHÓA 13 KHÓA NGÀY: 10 March 2020

HỆ THỐNG LÝ THUYẾT

1. **Khái niệm cơ bản**

****Một tiến trình gồm**

- ✓ Text section.
- ✓ Data section.
- ✓ Program counter.
- ✓ Process Status Word.
- ✓ Stack Pointer.
- ✓ Các thông tin quản lý bộ nhớ.

*****Các bước tạo tiến trình**

- ✓ Định danh cho tiến trình (duy nhất)
- ✓ Cấp phát không gian cho tiến trình
- ✓ Khởi tạo khối dữ liệu PCB cho tiến trình
- ✓ Thiết lập mối quan hệ cần thiết

2. **Trạng thái tiến trình**

```
/* Cho tiến trình c*/
int main(int argc, char** argv)
{
        Printf("Hello word");
        exit(0);
}
```

New
Ready
Running
Gặp printf -> waiting (chờ I/O)
Ready
Running -> Gặp Exit(0)
Terminal

Ví dụ 2: int main(int argv, char** argv)

```
{
        Int a,b,I;
        For(I =16; I >= 6;i--)
        {
                If(i%3 ==0)
                {
                        Printf("Số %d chi hết cho 3",i);
                }
else
                {
                        a = b + i;
                }
        }
        Exit(0);
}
```

Biểu diễn

New

Ready

Running

Ready -> Vì 16 không chia hết cho 3 nên xuống else, else không có printf nên không có

waiting

Running

Ready

Running

Waiting -> Vì lúc này 15/3 có lện printf

… tiếp tục đến khi nào hết vòng lặp xuất ra terminal.

3. Process Control Block (PCB)

- ✓ Mỗi tiến trình trong hệ thống đều được cung cấp một PCB.
- ✓ PCB là một cấu trúc dữ liệu quan trọng của hệ điều hành
- ✓ Một PCB gồm
  - Trạng thái tiến trình
  - Bộ đếm chương trình
  - Các thanh ghi
  - Thông tin lập lịch CPU
  - Thông tin quản lý bộ nhớ
  - Thông tin lượng CPU, thời gian sử dụng
  - Thông tin trạng thái I/O

4. Yêu cầu hệ điều hành đối với tiến trình

- ✓ Hỗ trợ thực thi luân phiên giữa các tiến trình.
- ✓ Phân phối tài nguyên hệ thống hợp lý.
- ✓ Tránh deadlock
- ✓ Cung cấp cơ chế giao tiếp và đồng bộ hoạt động giữa các tiến trình.
- ✓ Cung cấp cơ chế hỗ trợ user tạo/ kết thúc tiến trình.

5. Bộ định thời

- ✓ Bộ định thời công việc → Bộ định thời dài
- ✓ Bộ định thời CPU → Bộ định thời ngắn
- ✓ Các tiến trình có thể mô tả như: Hướng I/O, Hướng CPU
- ✓ Thời gian thực hiện khác nhau. Kết hợp hài hòa giữa chúng.
- ✓ Ngoài ra còn bộ định thời trung gian để điều chỉnh mức độ đa chương của hệ thống

6. Các tác vụ của tiến trình

- ✓ Tạo tiến trình mới: fork() thông qua lời gọi hệ thống.
- ✓ Tiến trình được tạo là con của tiến trình tạo(Tiến trình cha)
- ✓ Tiến trình con nhận tài nguyên từ hệ điều hành hoặc tiến trình cha.
- ✓ Chia sẻ tài nguyên
  - Tiến trình cha và con chia sẻ mọi tài nguyên
  - Tiến trình con chia sẻ một phần tài nguyên của tiến trình cha
- ✓ Trình tự thực thi
  - Cha và con thực thi đồng thời
  - Con thực thi kết thúc rồi đến cha (cha đợi con thực thi xong)
- ✓ Không gian địa chỉ
  - Không gian địa chỉ của tiến trình con được nhân bản từ cha
  - Không gian địa chỉ của tiến trình con được khởi tạo từ template.
- ✓ Kết thúc tiến trình: Exit
- ✓ Hệ điều hành thu tất cả tài nguyên khi chương trình kết thúc.

7. Cộng tác giữa các tiến trình

- ✓ Các tiến trình có thể cộng tác để hoàn thành công việc
- ✓ Mục tiêu cộng tác: Chia sẻ tài nguyên, tăng tốc tính toán, thực hiện công việc chung
- ✓ Cần sự hỗ trợ của hệ điều hành qua cơ chế giao tiếp và đồng bộ hđ của các tiến trình.

8. Tiểu trình

- ✓ Là đơn vị cơ bản sử dụng CPU
- ✓ Lợi ích của tiến trình đa luồng
  - Cho phép chương trình tiếp tục thực thi khi một bộ phận bị khóa hoặc một hoạt động dài.
  - Chia sẻ tài nguyên
  - Kinh tế.

9. Bài toán process với fork()

- ✓ **Số tiến trình được tạo ra = 2^ số fork tạo ra**
- ✓ **Số tiến trình con được tạo ra 2^ số fork tạo ra -1**

Ví dụ 1:

```
#include<stdio.h>

#include <string>

#include <unistd.h>

int main()

{fork ();

printf ("1 ");

fork ();

printf ("2 ");

fork ();

printf ("3 ");

return 0;}
```

Giải thích: Lệnh fork() đầu tiên tạo ra hai tiến trình con

Ví dụ 2: Int main(int argc,char * argv[1]){

```
Printf(“Hi”);
Int pid = fork();/*F1*/
If(pid>0){
        Fork();/*F2*/
        Printf(“Hello”);/*Pr1*/
}
Else
        Fork();/*F3*/
        Printf(“bye”);/*Pr2*/}
```

Ví dụ 3:

```
#include <stdio.h>
#include<unistd.h>
Int main (int argc, char *argv[])
{
int pid;
printf(“Hi”);
pid = fork();
if (pid >0)
{
        Fork();
        Fork();
        Printf(“Hello”);
}
Else
        Fork();
Printf(“Bye”);
}
```

BÀI TẬP

Câu 1. Cho đoạn mã chương trình sau. Có bao nhiêu trạng thái tiến trình waiting trong chương trình.

```c
#include <stdio.h>
int main ()
{
   int a = 10;
   do
   {
       if(a%2==0)
       {
           printf("Gia tri cua a la: %d\n", a);
           a++;
       }
       else
       {
           a = a -3;
       }
       a++;
   }while( a < 30 );
   return 0;
}
```

A. 1 **B.** 5 **C.** 10 **D.** 20

Câu 2. Cho đoạn chương trình sau. Có bao nhiêu chữ "Hello" được in ra?

```c
#include <stdio.h>
#include <unistd.h>
void main(int argc,char **argv)
{
    printf("Hi\n");
    int pid=fork();
    if(pid<0)
    {
        fork();
        printf("Hello\n");
    }
    else
        fork();
    printf("Bye\n");
        fork();
        printf("Hi\n");
}
```

A. 0 **B.** 8 **C.** 15 **D.** 10