

## Thao tác với Tiến Trình – Process

### Bài tập 1: Khởi tạo và Thu dọn Tiến trình

Khảo sát vòng đời cơ bản nhất của một tiến trình: được tạo ra, thực thi, kết thúc và được tiến trình cha chờ đợi.

#### Yêu cầu:

- Viết chương trình tạo một tiến trình con bằng `fork()`.
- Tiến trình cha: In ra PID của chính nó và PID của tiến trình con. Sau đó, sử dụng `wait()` để chờ con kết thúc. Dùng `WIFEXITED()` và `WEXITSTATUS()` để ghi nhận và in ra mã thoát của tiến trình con.
- Tiến trình con: In ra PID của chính nó, sau đó gọi `exit()` với một giá trị cụ thể (ví dụ: `exit(10)`).

### Bài tập 2: Thay thế Mã thực thi và Tương tác với Môi trường

Tìm hiểu cách một tiến trình có thể thay thế hoàn toàn mã lệnh đang chạy của nó bằng một chương trình khác và cách truyền thông tin qua biến môi trường.

#### Yêu cầu:

- Viết chương trình mà tiến trình cha thiết lập một biến môi trường (ví dụ: `MY_COMMAND=ls`).
- Tiến trình con sẽ đọc biến môi trường này. Dựa vào giá trị đọc được, nó sẽ dùng họ hàm `exec()` (ví dụ `execlp()`) để thực thi lệnh tương ứng (`ls`, `date`,...).
- Trong báo cáo, giải thích: Điều gì xảy ra với không gian địa chỉ và mã lệnh của tiến trình con sau khi `exec()` được gọi thành công?

### Bài tập 3: Khảo sát các Trạng thái Tiến trình Đặc biệt

Quan sát và phân tích hai trạng thái đặc biệt thường gặp trong quản lý tiến trình là Zombie và Orphan.

#### Yêu cầu:

- Tạo tiến trình Zombie:
  - Viết chương trình mà tiến trình con thoát ngay lập tức, nhưng tiến trình cha không gọi `wait()` mà `sleep()` trong một khoảng thời gian dài.
  - Sử dụng lệnh `ps` trong terminal để quan sát trạng thái `<defunct>` của tiến trình con.
- Tạo tiến trình Orphan:
  - Viết chương trình mà tiến trình cha thoát ngay sau khi tạo con.
  - Tiến trình con sẽ `sleep()` một lúc và trong thời gian đó, liên tục in ra PID của tiến trình cha (PPID). Quan sát sự thay đổi của PPID.
- Trong báo cáo, giải thích: Tại sao hai trạng thái này xuất hiện và ý nghĩa của chúng trong hệ thống Linux là gì?