Lab 2

Thực hành - Bài 2 : Lập lịch một tác vụ thời gian thực trên Raspberry Pi 4B/5

Mục tiêu

- Hiểu được khái niệm lập lịch thời gian thực trong Linux (PREEMPT_RT).
- Thực hành áp dụng các chính sách lập lịch: `SCHED_OTHER`, `SCHED_FIFO`.
- Quan sát ảnh hưởng của mức ưu tiên và scheduler đến độ ổn định của tiến trình.
- So sánh sự khác biệt giữa tác vụ thông thường và tác vụ thời gian thực trong môi trường có tải CPU cao.

Kiến thức nền tảng

- **`SCHED_OTHER`**: chính sách mặc định (Linux time-sharing).
- **`SCHED_FIFO`**: lập lịch thời gian thực kiểu First-In-First-Out.
- **`SCHED_RR`**: thời gian thực kiểu Round Robin (không sử dụng trong lab này).
- Ưu tiên (`priority`) trong `SCHED_FIFO` có giá trị từ **1–99**, càng cao càng được ưu tiên.

Yêu cầu chuẩn bị

- Raspberry Pi 4B/5 đã cài đặt kernel thời gian thực (PREEMPT_RT).
- Pi OS 64-bit (Bookworm hoặc tương đương).
- Trình biên dịch C++ (`g++`).
- Truy cập qua SSH hoặc terminal trực tiếp.

Nội dung thực hành

1. Tạo các file sau:

`hop.cpp` – tiến trình chiếm CPU

```cpp
int main() {
    while (true) { asm volatile("nop"); }
}
```

`task_normal.cpp` – tác vụ thông thường (SCHED_OTHER)

```cpp
#include <iostream>
#include <chrono>
#include <thread>
#include <cstdlib>   // std::atoi
#include <string>

int main(int argc, char* argv[]) {
    // --- Tham số mặc định ---
    std::string ten = "ten_SV";
    std::string mssv = "1233456789";
    int expected_interval = 100; // đơn vị: milliseconds
    int duration_minutes = -1;   // -1 nghĩa là chạy vô hạn

    // --- Đọc tham số dòng lệnh ---
    for (int i = 1; i < argc - 1; ++i) {
        std::string arg = argv[i];
        if (arg == "--interval") {
            expected_interval = std::atoi(argv[++i]);
            if (expected_interval <= 0) {
                std::cerr << "Loi: Khoang thoi gian (ms) khong hop le.\n";
                return 1;
            }
        } else if (arg == "--duration") {
            duration_minutes = std::atoi(argv[++i]);
            if (duration_minutes <= 0) {
                std::cerr << "Loi: Thoi gian chay (phut) khong hop le.\n";
                return 1;
            }
        }
    }

    std::cout << "=== Dang chay task_normal_monitor ===\n";
    std::cout << "Ho va ten: " << ten << "\n";
    std::cout << "mssv: " << mssv << "\n";
    std::cout << "Chu ky mong doi: " << expected_interval << " ms\n";
    if (duration_minutes > 0) {
        std::cout << "Thoi gian toi da: " << duration_minutes << "
phut\n";
    } else {
        std::cout << "Thoi gian chay: khong gioi han\n";
    }
```

```
    // --- Khởi tạo thời gian ---
    std::chrono::steady_clock::time_point start_time =
std::chrono::steady_clock::now();
    std::chrono::steady_clock::time_point prev_time = start_time;

    int prev_delta = -1;
    int change_count = 0;
    int total_loops = 0;

    // --- Vòng lặp chính ---
    while (true) {
        ++total_loops;
        std::chrono::steady_clock::time_point now =
std::chrono::steady_clock::now();
        int delta = std::chrono::duration_cast<std::chrono::milliseconds>
(now - prev_time).count();
        prev_time = now;

        if (delta != expected_interval) {
            ++change_count;
            long long now_ms =
std::chrono::duration_cast<std::chrono::milliseconds>
(now.time_since_epoch()).count();
            int elapsed_sec =
std::chrono::duration_cast<std::chrono::seconds>(now -
start_time).count();

            std::cout << "[Delta lệch] "
                      << "Thoi diem: " << now_ms << " ms | "
                      << "Chenh lech: " << delta << " ms (so voi " <<
expected_interval << " ms) | "
                      << "So lan lech: " << change_count << " | "
                      << "Thoi gian chay: " << elapsed_sec << " s\n";

            prev_delta = delta;
        }

        // Kiểm tra thời gian chạy giới hạn
        if (duration_minutes > 0) {
            auto elapsed =
std::chrono::duration_cast<std::chrono::minutes>(now -
start_time).count();
            if (elapsed >= duration_minutes) {
                std::cout << "\n=== Ket thuc chuong trinh ===\n";
                std::cout << "Tong vong lap: " << total_loops << "\n";
                std::cout << "So lan delta lech: " << change_count <<
"\n";
                std::cout << "Thoi gian chay: " << elapsed << " phut\n";
                break;
            }
```

```
        }


std::this_thread::sleep_for(std::chrono::milliseconds(expected_interval));
    }

    return 0;
}
```

`task_rt.cpp` – tác vụ thời gian thực (SCHED_FIFO)

```
#include <iostream>
#include <chrono>
#include <thread>
#include <cstdlib>
#include <string>
#include <sched.h>
#include <unistd.h>

int main(int argc, char* argv[]) {
    // --- Tham số mặc định ---
    std::string ten = "ten_SV";
    std::string mssv = "1233456789";
    int expected_interval = 100; // milliseconds
    int duration_minutes = -1;   // -1 = unlimited
    int priority = 80;           // SCHED_FIFO priority (1–99)

    // --- Đọc tham số dòng lệnh ---
    for (int i = 1; i < argc - 1; ++i) {
        std::string arg = argv[i];
        if (arg == "--interval") {
            expected_interval = std::atoi(argv[++i]);
            if (expected_interval <= 0) {
                std::cerr << "Loi: Khoang thoi gian khong hop le.\n";
                return 1;
            }
        } else if (arg == "--duration") {
            duration_minutes = std::atoi(argv[++i]);
            if (duration_minutes <= 0) {
                std::cerr << "Loi: Thoi gian chay khong hop le.\n";
                return 1;
            }
        } else if (arg == "--priority") {
            priority = std::atoi(argv[++i]);
            if (priority < 1 || priority > 99) {
                std::cerr << "Loi: Priority phai tu 1 den 99.\n";
                return 1;
            }
```

```
        }
    }

    // --- Thiết lập SCHED_FIFO ---
    sched_param sch;
    sch.sched_priority = priority;
    if (sched_setscheduler(0, SCHED_FIFO, &sch) == -1) {
        perror("sched_setscheduler");
        return 1;
    }

    std::cout << "=== Dang chay task_rt_monitor ===\n";
    std::cout << "Ho va ten: " << ten << "\n";
    std::cout << "mssv: " << mssv << "\n";
    std::cout << "Chinh sach: SCHED_FIFO | Uu tien: " << priority << "\n";
    std::cout << "Chu ky mong doi: " << expected_interval << " ms\n";
    if (duration_minutes > 0) {
        std::cout << "Thoi gian toi da: " << duration_minutes << "
phut\n";
    } else {
        std::cout << "Thoi gian chay: khong gioi han\n";
    }

    // --- Khởi tạo thời gian ---
    std::chrono::steady_clock::time_point start_time =
std::chrono::steady_clock::now();
    std::chrono::steady_clock::time_point prev_time = start_time;

    int prev_delta = -1;
    int change_count = 0;
    int total_loops = 0;

    while (true) {
        ++total_loops;
        std::chrono::steady_clock::time_point now =
std::chrono::steady_clock::now();
        int delta = std::chrono::duration_cast<std::chrono::milliseconds>
(now - prev_time).count();
        prev_time = now;

        if (delta != expected_interval) {
            ++change_count;
            long long now_ms =
std::chrono::duration_cast<std::chrono::milliseconds>
(now.time_since_epoch()).count();
            int elapsed_sec =
std::chrono::duration_cast<std::chrono::seconds>(now -
start_time).count();

            std::cout << "[Delta lệch] "
```

```
                      << "Thoi diem: " << now_ms << " ms | "
                      << "Chenh lech: " << delta << " ms (so voi " <<
expected_interval << " ms) | "
                      << "So lan lech: " << change_count << " | "
                      << "Thoi gian chay: " << elapsed_sec << " s\n";

            prev_delta = delta;
        }

        if (duration_minutes > 0) {
            auto elapsed =
std::chrono::duration_cast<std::chrono::minutes>(now -
start_time).count();
            if (elapsed >= duration_minutes) {
                std::cout << "\n=== Ket thuc chuong trinh ===\n";
                std::cout << "Tong vong lap: " << total_loops << "\n";
                std::cout << "So lan delta lech: " << change_count <<
"\n";
                std::cout << "Thoi gian chay: " << elapsed << " phut\n";
                break;
            }
        }


std::this_thread::sleep_for(std::chrono::milliseconds(expected_interval));
    }

    return 0;
}
```

2. Biên dịch tất cả chương trình:

```
g++ hop.cpp -o hop
g++ task_normal.cpp -o task_normal
g++ task_rt.cpp -o task_rt
```

3. Chạy các chương trình theo thứ tự sau:

a. Chạy các tác vụ thông thường:

```
./task_normal --interval 100 --duration 5
```

b. Mở terminal khác --> chạy 5 lần tiến trình chiếm CPU

```
./hop & ./hop & ./hop & ./hop & ./hop &
```

--> Quan sát `task_normal` bị trễ, mất nhịp in log.

c. Dừng task_normal, chạy tác vụ thời gian thực

```
sudo ./task_rt --interval 100 --duration 5 --priority 80
```

--> Quan sát log vẫn đều bất chấp tải cao, không xuất hiện trễ.

✅ **Ví dụ sử dụng:**

Mục tiêu	Lệnh
Chạy 5 phút, chu kỳ 100ms, priority 80	`sudo ./task_rt --interval 100 --duration 5 --priority 80`
Ưu tiên thấp hơn (50) để dễ bị ngắt	`sudo ./task_rt --interval 100 --priority 50`
Ưu tiên rất cao để đảm bảo đều	`sudo ./task_rt --interval 50 --priority 95`

Yêu cầu báo cáo

- Dán log đầu ra của `task_normal` trước và sau khi có `hop`
- Dán log của `task_rt` khi chạy cùng với `hop`
- Nhận xét:
  - So sánh độ ổn định giữa 2 tác vụ
  - Vì sao `task_rt` hoạt động tốt hơn
- Thử giảm `sched_priority` xuống 10,20,50 và ghi nhận thay đổi nếu có