

Lab

4

**BÁO CÁO BÀI THỰC HÀNH SỐ 4**

# **LẬP LỊCH TIẾN TRÌNH**

**Môn học: Hệ điều hành (IT007)**

<b>Sinh viên thực hiện</b>	Nguyễn Đức Tấn
<b>Thời gian thực hiện</b>	27/04/2024


**NỘI DUNG LÝ THUYẾT:**

Linux: Hệ điều hành đa nhiệm ưu tiên

Mỗi tiến trình sẽ được gán 1 độ ưu tiên

- + Theo arrival time (FCFS)
- + Burst time (SJF, SRTF)

<b>pn[]</b>	<b>arr[]</b>	<b>bur[]</b>	<b>star[]</b>	<b>finish[]</b>	<b>tat[]</b>	<b>wt[]</b>
Process name	Arrival time	Burst time	Start time	Finish time	Turn-around time	Waiting time
P0	0	3	arr[0]	star[0] + bur[0]	finish[0] - arr[0]	star[0] - arr[0]
P1	2	6	finish[0]	star[1] + bur[1]	finish[1] - arr[1]	star[1] - arr[1]
P2	4	1	finish[1]	star[2] + bur[2]	finish[2] - arr[2]	star[2] - arr[2]
P3	5	4	finish[2]	star[3] + bur[3]	finish[3] - arr[3]	star[3] - arr[3]
P4	6	2	finish[3]	star[4] + bur[4]	finish[4] - arr[4]	star[4] - arr[4]
...	...	...	...	...	...	...


  
**INPUT**

**Giải thuật SJF:**

```
#include <stdio.h>

void main() {
    int pn[10];
    int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], rt[10],
    i, j, n;
    int totwt = 0, tottat = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Process Name, Arrival Time & Burst Time for
Process %d: ", i + 1);
        scanf("%d %d %d", &pn[i], &arr[i], &bur[i]);
        rt[i] = bur[i]; // Khởi tạo thời gian thực thi còn lại ban đầu
        bằng burst time
    }

    int complete = 0; // Số tiến trình đã hoàn thành
    int time = 0; // Thời gian hiện tại

    while (complete != n) {
        int shortest = -1; // Chỉ số của tiến trình có burst time nhỏ
        nhất và đã đến
        int min_burst = 9999; // Burst time nhỏ nhất

        for (i = 0; i < n; i++) {
            if (arr[i] <= time && rt[i] < min_burst && rt[i] > 0) {
                shortest = i;
                min_burst = rt[i];
            }
        }

        if (shortest == -1) {
            time++;
            continue;
        }

        // Thực hiện tiến trình có burst time nhỏ nhất
        star[shortest] = time;
        time += rt[shortest];
        rt[shortest] = 0;
        finish[shortest] = time;
        tat[shortest] = finish[shortest] - arr[shortest];
        wt[shortest] = tat[shortest] - bur[shortest];
    }
}
```

```
        complete++;
    }

    printf("\nPName Arrival Burst Response Waiting Turnaround");
    for (i = 0; i < n; i++) {
        printf("\n%d\t%d\t%d\t%d\t%d\t%d", pn[i], arr[i], bur[i],
star[i], wt[i], tat[i]);
        totwt += wt[i];
        tottat += tat[i];
    }

    float avg_wt = (float) totwt / n;
    float avg_tat = (float) tottat / n;

    printf("\n\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f", avg_tat);
}
```

### Giải thuật SRTF:

```
#include <stdio.h>

void main() {
    int pn[10];
    int arr[10], bur[10], star[10], finish[10], tat[10], wt[10], rt[10],
i, j, n;
    int totwt = 0, tottat = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Process Name, Arrival Time & Burst Time for
Process %d: ", i + 1);
        scanf("%d %d %d", &pn[i], &arr[i], &bur[i]);
        rt[i] = bur[i]; // Khởi tạo thời gian thực thi còn lại ban đầu
bằng burst time
    }

    int complete = 0; // Số tiến trình đã hoàn thành
    int time = 0; // Thời gian hiện tại
    int min_burst = 9999; // Burst time nhỏ nhất

    while (complete != n) {
```

## Lab 4: Lập lịch tiến trình

```
int shortest = -1; // Chỉ số của tiến trình có burst time nhỏ nhất và đã đến

for (i = 0; i < n; i++) {
    if (arr[i] <= time && rt[i] < min_burst && rt[i] > 0) {
        shortest = i;
        min_burst = rt[i];
    }
}

if (shortest == -1) {
    time++;
    continue;
}

// Thực hiện một đơn vị thời gian cho tiến trình có burst time nhỏ nhất
rt[shortest]--;
min_burst = rt[shortest];

if (min_burst == 0) {
    min_burst = 9999; // Reset burst time nhỏ nhất

    complete++;
    finish[shortest] = time + 1;
    tat[shortest] = finish[shortest] - arr[shortest];
    wt[shortest] = tat[shortest] - bur[shortest];

    totwt += wt[shortest];
    tottat += tat[shortest];
}

time++;
}

printf("\nPName Arrival Burst Response Waiting Turnaround");
for (i = 0; i < n; i++) {
    printf("\n%d\t%d\t%d\t%d\t%d\t%d", pn[i], arr[i], bur[i], star[i], wt[i], tat[i]);
}

float avg_wt = (float) totwt / n;
float avg_tat = (float) tottat / n;

printf("\n\nAverage Waiting Time: %.2f", avg_wt);
printf("\nAverage Turnaround Time: %.2f", avg_tat);
}
```

**Giải thuật Round Robin:**

```

#include <stdio.h>

void main() {
    int pn[10];
    int burst[10], rem_burst[10], wt[10], tat[10], i, n, quantum;
    int totwt = 0, tottat = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    printf("Enter the quantum time: ");
    scanf("%d", &quantum);

    for (i = 0; i < n; i++) {
        printf("Enter Process Name and Burst Time for Process %d: ", i +
1);
        scanf("%d %d", &pn[i], &burst[i]);
        rem_burst[i] = burst[i]; // Khởi tạo thời gian thực thi còn lại
        ban đầu bằng burst time
    }

    int complete = 0; // Số tiến trình đã hoàn thành
    int time = 0; // Thời gian hiện tại

    while (complete != n) {
        for (i = 0; i < n; i++) {
            if (rem_burst[i] > 0) {
                if (rem_burst[i] <= quantum) {
                    time += rem_burst[i];
                    wt[i] = time - burst[i];
                    tat[i] = time;
                    rem_burst[i] = 0;
                    complete++;
                } else {
                    time += quantum;
                    rem_burst[i] -= quantum;
                }
            }
        }
    }

    printf("\nGantt Chart:");
    printf("\nProcess\tStart Time\tStop Time");
    for (i = 0; i < n; i++) {

```

```
        printf("\n%d\t%d\t\t%d", pn[i], wt[i], tat[i]);
        totwt += wt[i];
        tottat += tat[i];
    }

    float avg_wt = (float) totwt / n;
    float avg_tat = (float) tottat / n;

    printf("\n\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f", avg_tat);
}
```

