Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Операционные системы

Лабораторная работа 2

"Планирование процессов"

Вариант 9

Выполнил: Шкаруба Н.Е.

Группа: P3318

2016 г

Исходные данные:

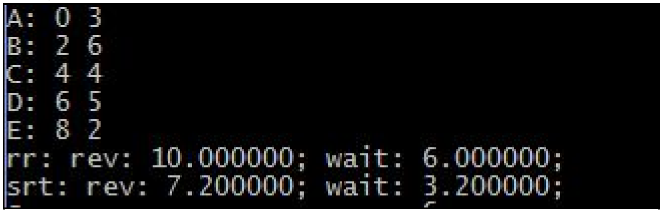
Процесс	Время запуска	Время обслуживания
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Код:

```
float rr(std::vector<ProcInfo> p_info, int quantum){
        _log("\nrr:\n\n");
        std::deque<ProcInfo> proc_start_queue(p_info.begin(), p_info.end());
        std::deque<ProcInfo> proc_work_queue;
        ProcInfo proc, saved_proc;
        int TR = 0,
                real_quantum = 0;
        bool need_more_time = false;

        while (!(proc_start_queue.empty() && proc_work_queue.empty())){
                while (!proc_start_queue.empty()){
                        proc = proc_start_queue.front();
                        if (TR < proc.start_time) break;

                        proc_start_queue.pop_front();
                        proc_work_queue.push_back(proc);
                        _log("TR = %d\tnew proc(%c)\n", TR, proc.id + 'A');
                }

                for (int i = 0; i<proc_work_queue.size(); i++){
                        _log("<-%c->", proc_work_queue[i].id + 'A');
                        if (p_info[proc_work_queue[i].id].wait_time == 0){
        p_info[proc_work_queue[i].id].wait_time =TR -
p_info[proc_work_queue[i].id].start_time;
                        } else{
                                p_info[proc_work_queue[i].id].wait_time += real_quantum;
                        }
                }
                if (proc_work_queue.size() > 0)
                        _log("\n", proc_work_queue.size());

                if (need_more_time){
                        need_more_time = false;
                        proc_work_queue.push_back(saved_proc);
                }

                TR += quantum;
                if (proc_work_queue.empty()) continue;

                proc = proc_work_queue.front();
                proc_work_queue.pop_front();
                proc.work_time -= quantum;

        _log("TR = %d\tproc(%c): %d %d\n", TR, proc.id + 'A', proc.start_time,
proc.work_time);

                if (proc.work_time > 0){
                        need_more_time = true;
                        saved_proc = proc;
                        real_quantum = quantum;
                } else{
                        TR += proc.work_time;
                        real_quantum = quantum + proc.work_time;
        }
    }

    float avg_wait = 0, avg_rev = 0;

    for (int i = 0; i<p_info.size(); i++){
        proc = p_info[i];
_log("%c\tstart: %d; work: %d; wait: %d; re: %d\n", proc.id + 'A', proc.start_time,
        proc.work_time, proc.wait_time, proc.work_time + proc.wait_time);
        avg_wait += proc.wait_time;
        avg_rev += proc.work_time + proc.wait_time;
    }

    avg_rev /= (float)p_info.size();
    avg_wait /= (float)p_info.size();

    printf("rr: rev: %f; wait: %f;\n", avg_rev, avg_wait);
    return avg_rev;
}
```

```
float srt(std::vector<ProcInfo> p_info){
    _log("\nsrt:\n\n");
    std::deque<ProcInfo> proc_start_queue(p_info.begin(), p_info.end());
    std::deque<ProcInfo> proc_work_queue;
    int quantum = 1;
    ProcInfo proc;
    int TR = 0,
        time_left = INT_MAX,
        min_time,
        min_time_proc_id,
        cur_working_proc_id = 0;

    while (!(proc_start_queue.empty() && proc_work_queue.empty())){
        while (!proc_start_queue.empty()){
            proc = proc_start_queue.front();
            if (TR < proc.start_time) break;

            proc_start_queue.pop_front();
            proc_work_queue.push_back(proc);

p_info[proc_work_queue[cur_working_proc_id].id].wait_time +=TR -
p_info[proc_work_queue[cur_working_proc_id].id].last_work_time;

            if (proc.work_time < time_left){
                cur_working_proc_id = proc_work_queue.size() - 1;
            }
_log("TR = %d\tnew proc(%c)\n", TR, proc_work_queue[proc_work_queue.size() - 1].id + 'A');
        }

        TR += quantum;
        if (proc_work_queue.empty()) continue;

        if (time_left <= 0){
            _log("we change working proc due to end\n");
            if (proc_work_queue[cur_working_proc_id].work_time <= 0)
            proc_work_queue.erase(proc_work_queue.begin() + cur_working_proc_id);

            if (proc_work_queue.empty()) continue;
            min_time = proc_work_queue[0].work_time;
            min_time_proc_id = 0;
            for (int i = 1; i<proc_work_queue.size(); i++){
                proc = proc_work_queue[i];
                if (min_time > proc.work_time){
                    min_time = proc.work_time;
                    min_time_proc_id = i;
                }
            }
            cur_working_proc_id = min_time_proc_id;
        }

        proc = proc_work_queue[cur_working_proc_id];
        time_left = proc.work_time - quantum;
        proc.work_time = time_left;
        proc_work_queue[cur_working_proc_id] = proc;

_log("TR  = %d\tworking proc(%c): %d %d %d\n", TR, proc.id + 'A', proc.start_time,
        proc.work_time, p_info[proc.id].wait_time);

        p_info[proc_work_queue[cur_working_proc_id].id].last_work_time = TR;

        for (int i = 0; i<proc_work_queue.size(); i++){
            if (i == cur_working_proc_id) continue;
            _log("<-%c->", proc_work_queue[i].id + 'A');
            p_info[proc_work_queue[i].id].wait_time += quantum;
        }
        if (proc_work_queue.size() > 1) _log("\n");
    }

    float avg_wait = 0, avg_rev = 0;

    for (int i = 0; i<p_info.size(); i++){
        proc = p_info[i];
_log("%c\tstart: %d; work: %d; wait: %d; re: %d\n", proc.id + 'A', proc.start_time,
    proc.work_time, proc.wait_time, proc.work_time + proc.wait_time);
        avg_wait += proc.wait_time;
        avg_rev += proc.work_time + proc.wait_time;
    }

    avg_rev /= (float)p_info.size();
    avg_wait /= (float)p_info.size();

    printf("srt: rev: %f; wait: %f;\n", avg_rev, avg_wait);
    return avg_rev;
}
```

Результат:



```
A: 0 3
B: 2 6
C: 4 4
D: 6 5
E: 8 2
rr: rev: 10.000000; wait: 6.000000;
srt: rev: 7.200000; wait: 3.200000;
```