EcoLab4

Реализация алгоритма планирования **«LJF»**

Выполнил

Семенов Никита

21ПИ-2

Нижний Новгород

2024

Оглавление

[Описание алгоритма 3](#_Toc168240681)

[Реализация 4](#_Toc168240682)

[QEMU 4](#_Toc168240683)

# Описание алгоритма

**Longest Job First (LJF) — это** алгоритм планирования задач, в котором процессор всегда выбирает для выполнения ту задачу, которая требует наибольшего времени для завершения. Этот алгоритм противоположен алгоритму **"Shortest Job First" (SJF)**, который отдает приоритет задачам с наименьшим временем выполнения.

**Плюсы**

1. Минимизация переключений контекста: так как длинные задачи выполняются реже, уменьшается количество переключений между задачами, что снижает накладные расходы.
2. Оптимально для определенных типов нагрузок: В сценариях, где приоритет отдаётся более длительным и критическим задачам, LJF может быть полезным.

**Минусы**

1. Увеличение времени отклика: Короткие задачи могут ждать очень долго, что приводит к увеличению среднего времени отклика и задержек.
2. Несправедливость по отношению к коротким задачам: Короткие задачи могут испытывать "голодание", если постоянно добавляются более длинные задачи.
3. Требование точного знания времени выполнения задач: необходимо заранее знать или точно оценивать время выполнения каждой задачи, что не всегда возможно.
4. Меньшая гибкость: В реальных системах задачи могут иметь разные приоритеты, и выполнение самых длинных задач может не всегда быть оптимальным.
5. В общем, LJF может быть полезным в определенных специфических сценариях, но из-за его недостатков в реальных системах он используется редко.

# Реализация

void ChangeTaskOrder(CEcoTaskScheduler1Lab\_C761620F\* pCMe, uint64\_t\* task\_order, uint16\_t task\_count) {

uint64\_t i, j;

uint64\_t real\_task\_count;

uint64\_t tmp\_task;

real\_task\_count = task\_count;

for (i = 0; i < real\_task\_count - 1; ++i) {

for (j = i+1; j < real\_task\_count; ++j) {

if (pCMe->m\_pTaskList[task\_order[i]].runtime < pCMe->m\_pTaskList[task\_order[j]].runtime) {

tmp\_task = task\_order[i];

task\_order[i] = task\_order[j];

task\_order[j] = tmp\_task;

}

}

}

}

# QEMU

