Задание 1. Запуск процессов по расписанию.

Требуется написать программу `mycron`, которая будет исполнять программы в заданное время. Информация о времени и запущенных программах хранится в файле `mycrontab`.

Формат файла следующий:

```
10:30:30 cat /usr/include/stdio.h
*:00:00 echo beep
*:*:0 ls -l /
```

В каждой строке присутствует строка времени (разделённая двоеточиями на часы, минуты и секунды). В первом и втором поле времени могут быть звёздочки, означающие, что часы (звёздочка в первой позиции) и минуты (звёздочка во второй позиции) будут любыми, например,

```
*:*:0 ls -l /
```

должно приводить к запуску `ls -l /` ровно в 0 секунд каждой минуты.

Файл `mycrontab` может изменяться при работе программы (например, вы его можете отредактировать для того, чтобы запустить другую программу). Если добавились новые задачи или удалены старые — требуется завершить все исполняющиеся процессы и запустить всё по-новому.

Требуется самостоятельно разобрать строки во входном файле и правильно и вовремя запускать нужные программы. Нельзя использовать функции `system`, `popen` и аналогичные, а так же вызовы оболочки с передачей ей строки выполнения. Все аргументы должны быть разобраны самостоятельно.

Количество строк во входном файле может быть очень большим, длина любой из строк может быть очень большой.

Последний срок сдачи задачи №1 — 4 ноября.

Задание 2. Параллельное копирование файлов.

Требуется написать программу параллельного копирования файлов. В командной строке задаётся количество потоков (`-t4`), копирующих файлы, директория-источник и директория-приёмник рекурсивно. Программа должна породить нужное количество потоков только один раз (нельзя для каждого копирования порождать новый поток!). Если копируемый файл `name` уже существует в директории-приёмнике, то его надо переименовать в `name.old` (если такой уже существовал, он затирается). Если директория-приёмник не существовала, её надо создать. Все права на файлы и их даты создания и модификации должны быть скопированы.

Пример вызова:

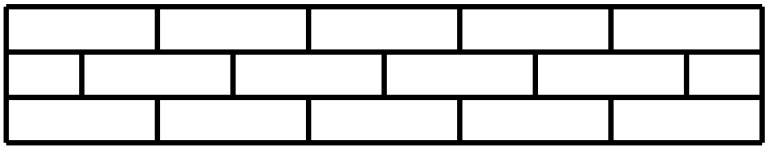
```
./pcopy -t4 /usr/include myincludes
```

Максимальную оценку получает решение, которое старается задействовать все потоки пополной. В идеальном решении простаивания потоков не должно быть или оно должно быть как можно меньшим.

Последний срок сдачи задачи №2 — 25 ноября.

Задание 3. Строители.

Отец и сын решили построить дом из кирпича. Каждая стена состоит из двух типов рядов: один начинается с целого количества кирпичей, другой с половины кирпича.



Положить кирпич в следующий ряд можно при условиях, что он

1. первый в ряду или примыкает к другому кирпичу из ряда;
2. полностью лежит на кирпичах из нижнего ряда.

Требуется написать программу для параллельного построения стены двумя потоками, отцовским и сыновним. Каждый кирпич укладывается случайное время, линейно распределённое между заданными границами, для отца и сына отдельно. Нижний ряд выкладывает всегда отец. Количество рядов всегда чётно. Кирпичи не должны падать и участники не должны блокировать друг друга. Второй и последующие чётные ряды начинаются и заканчиваются половинками кирпича. Должна быть минимальная визуализация процесса построения, вполне достаточно вывода текста с происходящими событиями и схемой уже построенной стены.

Рекомендуется использовать библиотеку `pthread`.

Формат входных данных:

```
N M
PMIN PMAX
SMIN SMAX
```

- N - ширина стены в целых кирпичах
- M - высота стены в кирпичах
- PMIN,PMAX — наименьшее и наибольшее время выкладывания кирпича отцом
- SMIN,SMAX — наименьшее и наибольшее время выкладывания кирпича сыном

Последний срок сдачи задачи №3 — 9 декабря.