САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет компьютерных технологий и управления
Кафедра вычислительной техники
Направление подготовки(специальность) 230100

О Т Ч Е Т
по практике

Тема задания: тема

Студенты Вакс Алексей, группа 3103

Руководитель практики Соснин В.В.

Оценка руководителя

Дата

Санкт-Петербург
2013г.

1 TeX(LaTeX)

TEX - система компьютерной вёрстки, разработанная Дональдом Кнутом в целях создания компьютерной типографии. В отличие от визуальных редакторов, TEX сам форматирует документ на основе выбранного пользователем шаблона — как правило, хранящегося в файле формата .tex. Далее файлы .tex транслируются в файлы .dvi, которые уже могут быть напечатаны либо перегнаны в формат pdf.

TEX/ предназначен для набора научного текста, на большую часть состоящего из формул, и обеспечивает удобное форматирование и оформление текста, перекрестных ссыло и библиографии, а также позволяет создавать более качественные сложные документы, чем WYSIWYG-редакторы — такие как MS Word.

Документы набираются на собственном низкоуровневом языке разметки TEX, содержащем команды отступа и смены шрифта, а также расширением TEX'а — LATEX'ом, который содержит в себе набором готовых стилей и шаблонов, позволяющих, например, быстро вставлять оглавнение, не задумываясь о нумерации отдельных объектов, что в MS Word гораздо сложнее.

Главным преимуществом TEX является возможность автоматизировать форматирование и структурирование документа, а также более гибкие возможности набора сложных формул, а также обеспечение кроссплатформенности и совместимости с PS и PDF.

Главным недостатком TEX является сложность в освоении и адаптации после WYSIWYG-редакторов, а также некоторые затруднения при работе с графикой, так как необходимо создавать для этого .eps-файл.

1.1 MiKTeX

MiKTeX - открытый дистрибутив TEX для платформы Windows, имеющий возможность автоматического обновления установленных компонентов и пакетов.

В состав MiKTeX включены:

- классический TEX-компилятор;
- различные варианты TeX: pdfTeX, e-TeX, pdf-e-TeX, и т.д.;
- конверторы TeX в PDF;
- MetaPost — интерпретатор для графических иллюстраций;
- полный набор общеиспользуемых макропакетов: LaTeX, ConTeXt и др.;
- средство просмотра Yap;
- инструменты и утилиты;

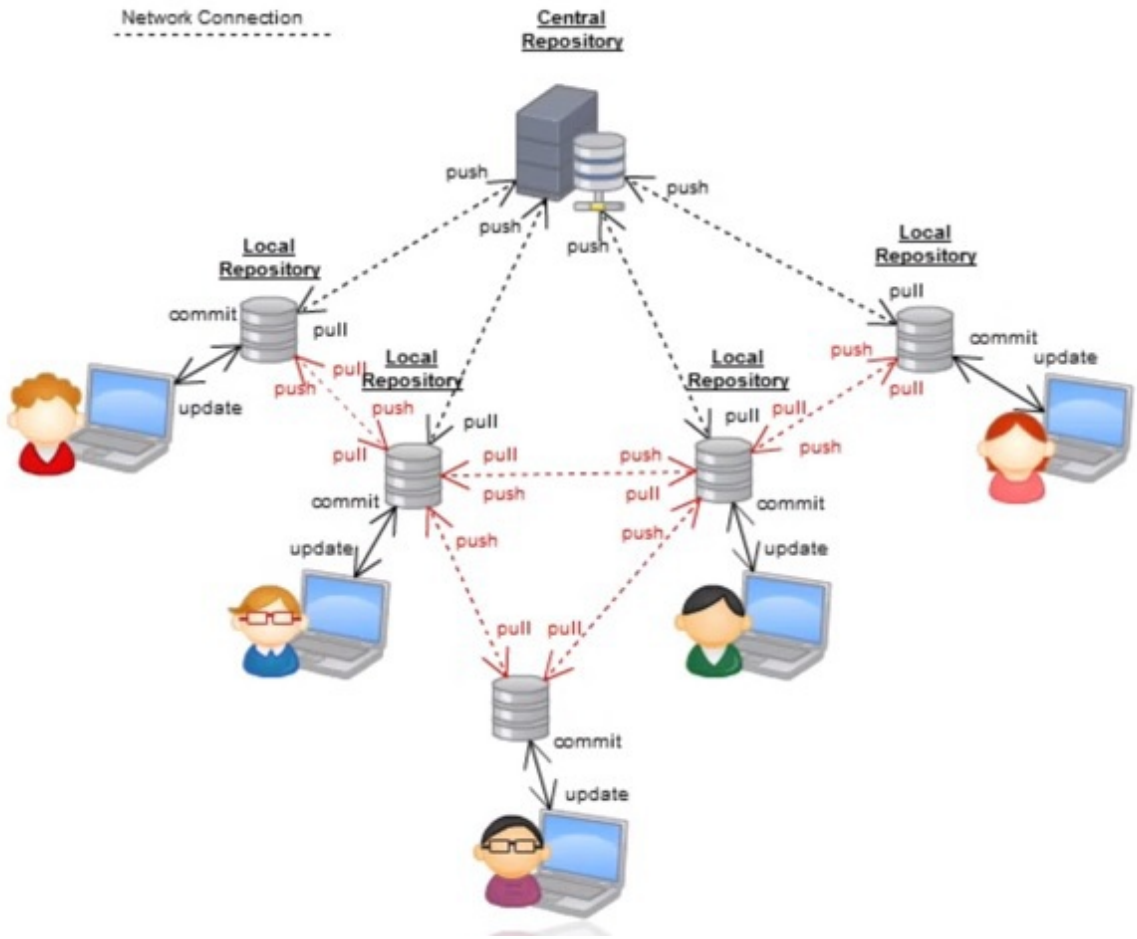
2 Git

Git - система управления версиями файлов, предназначенная для хранения, разделения и слияния версий и сохранения всей истории разработки в репозитории.

Для сохранения изменений используется команда commit.

Система контроля версий необходима для того, чтобы каждый из группы разработчиков видел свои собственные изменения, а также изменения, сделанные его коллегами, чтобы организовать быструю замену нужных файлов для замены всех модулей на модули последних версий, а также для возможностей возврата к предыдущим.

В Git между главным репозиторием и пользователем существует промежуточный репозиторий — локальный, который обеспечивает работу системы без интернета.

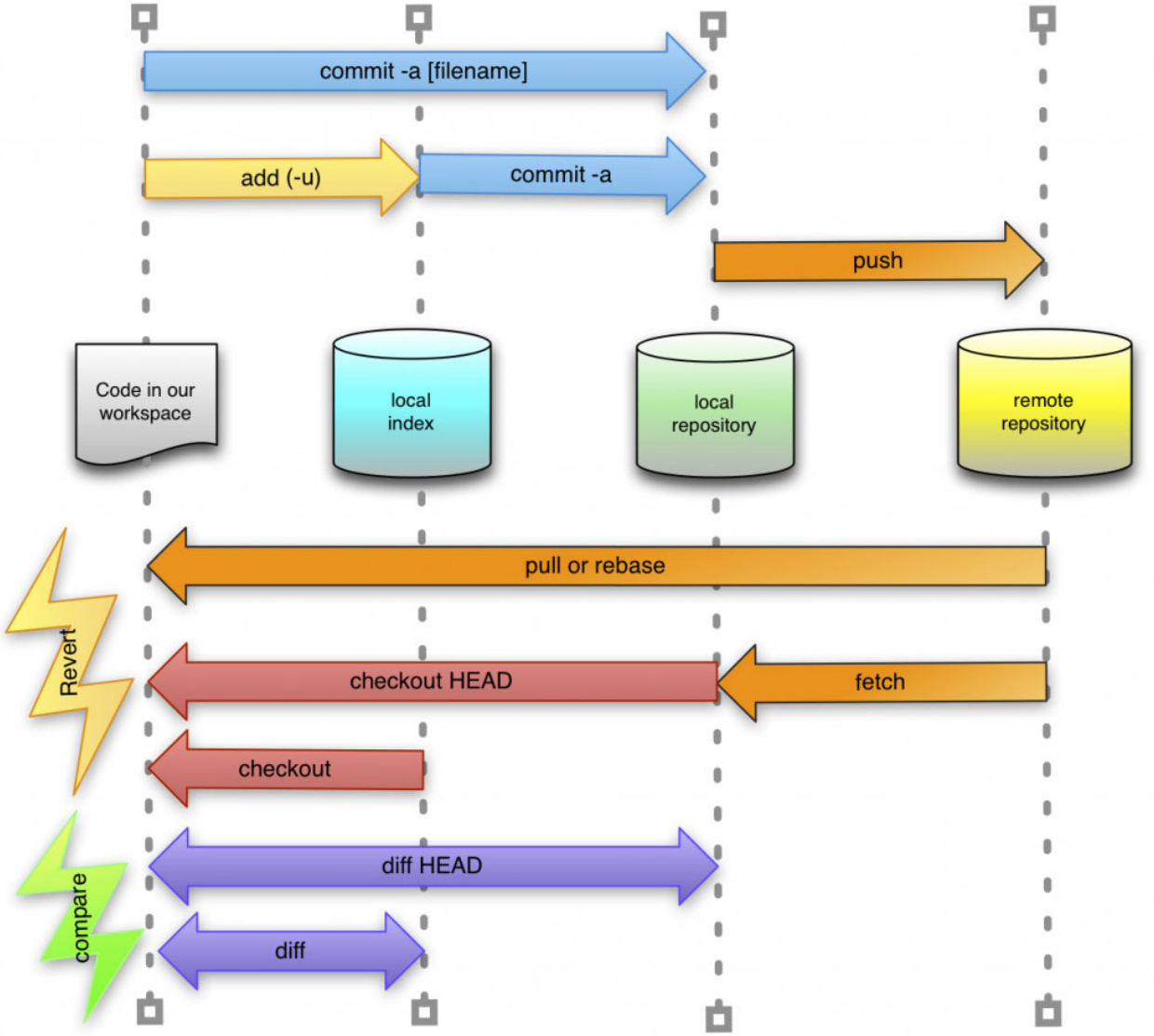


2.1 Описание работы Git:

Сотрудник работает в своем рабочем пространстве (workspace), система параллельно с его изменениями вносит изменения в главный индекс, следя за теми файлами, которые добавил сотрудник. После этого сотрудник может сохранить текущее состояние. Далее файлы добавляются в локальный репозиторий.

При наличии интернета сотрудник может перенести файлы в удаленный репозиторий (например, github). После этого из удаленного репозитория можно загрузить все файлы с текущими изменениями, а также просмот-

реть изменения текущей версии. Кроме того, другие пользователи также могут одновременно загружать свои изменения.



2.2 Основные git-команды:

git init - создание нового каталога с файлами нового репозитория.

git commit - сохранение изменений.

git status - вывод информации обо всех изменениях, внесенных в дерево директорий проекта по сравнению с последним коммитом рабочей ветки.

git add - ввод в индекс (временное хранилище) изменений, которые затем войдут в commit.

Основные ключи, используемые для упрощения работы с git:

git commit -a - выполняет commit, индексируя изменения в файлах проекта без индексации новых файлов, но с учитыванием удаления.

git commit -m «commit comment» - комментируем commit из командной строки вместо текстового редактора.

git commit "filename" - вносит в индекс и создаёт commit на основе изменений только одного файла.

git reset - возврат к определенному commit'у, откат изменений, «жесткий» или «мягкий». Мягкий оставляет индекс и дерево файлов и директорий нетронутым, а жёсткий возвращает указанное состояние, удаляя все последующие изменения, вызванные commit'ами.

git diff - просмотр изменений, не внесенных в индекс.

git push - внесение изменения в удаленный репозиторий.

git pull - возвращает изменения из удаленного репозитория.

Использованные материалы: http://git-scm.com/, http://robotics.usc.edu/.

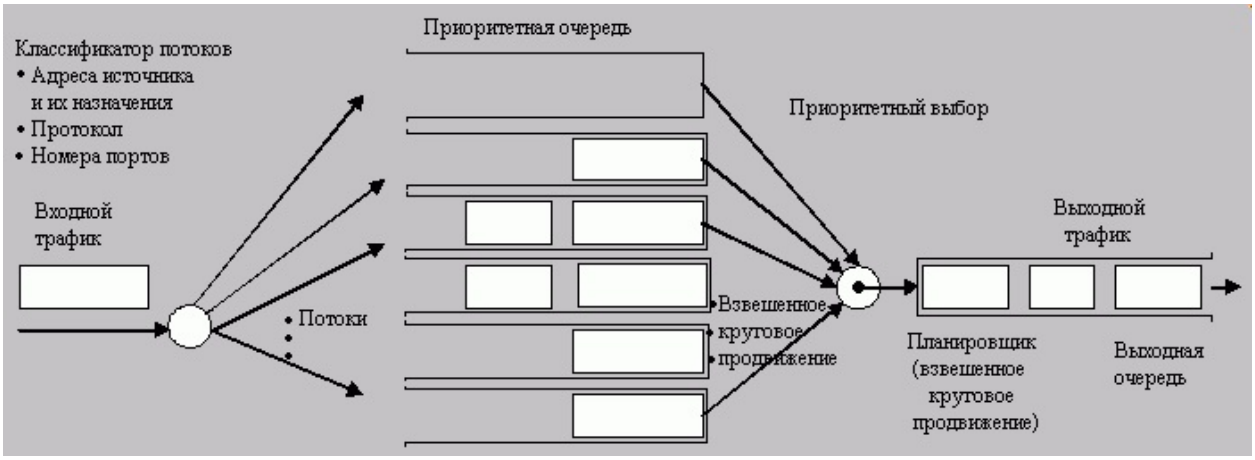
3 Взвешенная справедливая очередь

Взвешенная справедливая очередь (англ. Weighted fair queuing, WFQ) — механизм планирования пакетных потоков данных с различными приоритетами. Его целью является регулирование использования одного канала передачи данных несколькими конкурирующими очередями пакетов данных.

Взвешенная справедливая очередь – это комбинированный механизм, сочетающий приоритетное обслуживание с взвешенным и используя принципы и того и другого. От приоритетного обслуживания взята возможность устанавливать препочтение одним очередям пакетов данных, а от взвешенного обслуживания взята возможность предоставлять всем очере-

дям пакетов заявок определенный минимум пропускной способности, чтобы гарантировать некоторые требования к величине задержек.

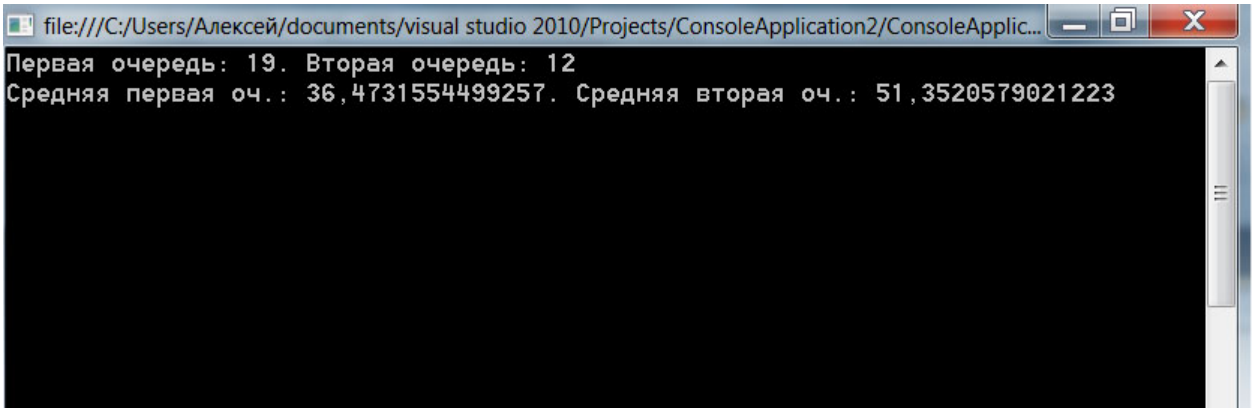
Типичный вариант реализации WFQ, когда всем классам трафика достаются равные доли пропускной способности выходного интерфейса из оставшейся от трафика приоритетного класса доли:



3.1 Реализация WFQ на языке программирования

Напишем программу, эмулирующую работу системы массового обслуживания по механизму WFQ с экспоненциальным распределением размеров пакетов (т.е. длительности обслуживания) и экспоненциальным потоком пакетов на языке C-Sharp в консольном приложении.

Результат работы программы и код:



```
        public static int numlast(double[] arr)
        {
            int ans = arr.Length;
            for (int i = 0; i < arr.Length; i++)
                if (arr[i] == 0)
                {
                    ans = i; //возврат номера последнего свободного места в очереди
                    break;
                }
            return ans;
        }
        static void Main(string[] args)
        {
            double timer = 0.08; //минимальная единица времени
            double nexts = 0; //переменная для сдвига массива
            double progress = 0; //переменная для обеспечения возможности ситуации, когда одна заявка не успевает обр
            //переменная для отслеживания соотношения распределения времени между обслуживанием разных классов заявок
            double buf1m = 0, buf2m = 0; //расчет средней длины очереди
            double buf1ms = 0, buf2ms = 0;
            int kol = 0;
            int buf1size = 280, buf2size = 180;
            int buf1 = 0, buf2 = 0;
            double[] arr1 = new double[buf1size];
            double[] arr2 = new double[buf2size];
            double servtime1 = 0.2, servtime2 = 0.7;
            double intensity1 = 4.1, intensity2 = 2.2;
            double kpd = 0;
            double load = 0;
            double SWITCHINGFINAL = 5; //гибкое переключение на другую очередь после обработки определенного кол-ва з
            double SWITCHINGPROGRESS = 0;
            double SWITCHING = 1;
            //обнуление массивов
            for (int i = 0; i < buf1size; i++)
            {
                arr1[i] = 0;
            }
            for (int i = 0; i < buf2size; i++)
            {
                arr2[i] = 0;
            }
            //
            //Главный цикл работы системы:
            //
            for (double i = timer; i < 100000; i += timer)
            {
                //добавляем новые заявки в очередь
                kol = Convert.ToInt32(Gen(timer * intensity1));
                for (int j = 0; j < kol; j++)
                {
                    if (ifnotfull(arr1))
                    {
                        arr1[numlast(arr1)] = Gen(servtime1);
                    }
                }
                kol = Convert.ToInt32(Gen(timer * intensity2));
                for (int j = 0; j < kol; j++)
                {
                    if (ifnotfull(arr2))
                    {
                        arr2[numlast(arr2)] = Gen(servtime2);
                    }
                }
                //закончили добавлять новые заявки, теперь обслуживаем заявки, проверка переключения

                if (SWITCHINGPROGRESS >= SWITCHINGFINAL)
                {
                    if (SWITCHING == 1) SWITCHING = 2; else SWITCHING = 1;

                    SWITCHINGPROGRESS = 0;
                }
```

```
                //обслуживание заявок
                if (SWITCHING == 1)
                {
                    if ((arr1[0] > 0))
                    {
                        arr1[0] -= servtime1;
                        SWITCHINGPROGRESS += servtime1;
                    }
                    else SWITCHING = 2;
                }

                if (SWITCHING == 2)
                {
                    if ((arr2[0] > 0))
                    {
                        arr2[0] -= servtime2;
                        SWITCHINGPROGRESS += servtime2;
                    }
                    else SWITCHING = 1;
                }

                //закончили обслуживать заявки, теперь сдвигаем массив

                if ((arr1[0] <= 0))
                {
                    for (int j = 0; j < buf1size-1; j++)
                    {
                        arr1[j] = arr1[j + 1];
                    }
                    arr1[buf1size - 1] = 0;
                }

                if ((arr2[0] <= 0))
                {
                    for (int j = 0; j < buf2size - 1; j++)
                    {
                        arr2[j] = arr2[j + 1];
                    }
                    arr2[buf2size - 1] = 0;
                }

                //подготавливаем информацию для вывода
                buf1 = numlast(arr1);
                buf2 = numlast(arr2);
                buf1ms += buf1;
                buf2ms += buf2;
                buf1m = buf1ms / (i / timer);
                buf2m = buf2ms / (i / timer);

                Console.Clear();
                Console.WriteLine("Первая очередь: " + buf1 + ". Вторая очередь: " + buf2 + "\t");
                Console.WriteLine("Средняя первая оч.: " + buf1m + ". Средняя вторая оч.: " + buf2m + "\t");
                //Console.WriteLine("Загруженность прибора: " + kpd + ". Нагрузка на прибор: " + load + "\t");
                Console.WriteLine("SWITCHING: " + SWITCHING + ". PROGRESS: " + SWITCHINGPROGRESS + "\t");

                Thread.Sleep(1);
            }
            Console.ReadLine();
        }
    }
}
```