СРАВНЕНИЕ ОСНОВНЫХ ДИСЦИПЛИН ПЛАНИРОВАНИЯ В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

В. П. Дашевский

Санкт-Петербургский институт информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия В.О., д.39
hoodwin@mail.ru

УДК 681.3.06:32
В. П. Дашевский. **Сравнение основных дисциплин планирования в системах реального времени** // Труды СПИИРАН. Вып. 1, т. 3. — СПб: СПИИРАН, 2003.
Аннотация. *Рассмотрены три дисциплины планирования: дисциплина статических приоритетов, дисциплина разделения времени и дисциплина ближайшего срока. Проведено сравнение их характеристик при планировании задач реального времени. Предложен новый, более простой способ доказательства возможности полного использования процессорного ресурса без пиковых перегрузок при использовании дисциплины ближайшего срока.* — Библ. 6 назв.

UDC 681.3.06:32
V. P. Dashevsky. **Comparing several scheduling algorithms for real-time systems** // SPIIRAS Proceedings. Issue 1, v.3. — SPb: SPIIRAS, 2003.
Abstract. *Three scheduling algorithms are examined, static priority-driven scheduling, time sliced scheduling and deadline-driven sheduling. Their characteristics are compared for scheduling tasks in real time. New, more simple proof is offered to show that deadline-driven scheduling algorithm is capable to utilize all proocessor resource without transient overloads*. — Bibl. 6 items.

1. Введение

Среди задач, решаемых при помощи ЭВМ, принято выделять отдельный класс задач, называемых задачами реального времени.

Определение. *Задачей реального времени называется задача, решение которой должно быть завершено к определенному, заранее фиксированному моменту времени.*

Важно сразу заметить интересное свойство этого определения. В нем отсутствует указание на характерный масштаб времени. Как будет видно из дальнейшего описания, это определение одинаково применимо и к задачам микросекундных масштабов, и к объемным задачам на целые пятилетки.

Обычно считается, что задачи реального времени предъявляют жесткие требования к организации ЭВМ. Именно поэтому их и выделяют в отдельный класс задач. Ниже будет показано, что общий принцип, предназначенный для задач реального времени, так же хорошо применим к любым задачам вообще, даже к таким, для решения которых ЭВМ вовсе не требуется.

Если ЭВМ приходится решать всего одну задачу реального времени, и других задач не имеется вовсе, то такой случай не представляет большого интереса, потому что, очевидно, вся трудность реального времени упирается в необходимость обеспечить достаточную производительность вычислительной машины. Крайне желательно при этом обеспечить еще некоторый временной запас для большей надежности, поскольку довольно часто алгоритмы имеют ветвления, так что невозможно точно оценить время их работы. Учитывая, что производительность ЭВМ постоянно совершенствуется, зачастую оказывается,

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

что временной запас уже настолько велик, что можно пытаться загрузить ЭВМ сразу несколькими задачами для большей эффективности её использования. Однако, как только перед ЭВМ ставится сразу несколько задач, возникает потребность в *дисциплине планирования,* такой, чтобы все задачи успевали решаться к отведенному им моменту времени. Особенно ярко это проявляется в ситуациях, когда перед ЭВМ (помимо прочих задач) одновременно ставится несколько задач реального времени. В ситуации нескольких задач будем говорить об ЭВМ как о *системе задач реального времени,* или, кратко, о *системе реального времени.*

Практически очень часто потребность в решении задачи реального времени возникает с поступлением в ЭВМ сигнала из внешней среды. Такие сигналы оформляются в виде *прерываний,* которые выдаются вычислительному устройству, асинхронным образом информируя его о том, что внешнее событие произошло. В частности, одним из таких событий может служить прерывание от таймера. Если вся система управляется только событиями от таймера, то в некоторых ситуациях существует возможность упростить дисциплину планирования по сравнению с общим случаем. Такой подход применяется в некоторых специализированных системах, но в данной статье такие системы задач будут рассматриваться в классе систем общего вида, то есть по отношению к более общим дисциплинам планирования.

Когда происходит прерывание, то ставится задача об обработке этого прерывания. В случае, когда эта задача является задачей реального времени, момент времени, к которому задача должна быть решена, определяется как текущий момент (возникновения прерывания) плюс некоторое время, за которое прерывание должно быть обработано. Это время назначается разработчиком системы в соответствии со спецификой решаемой задачи. По сути дела это время представляет собой величину, обратную максимальной частоте возникновения этого прерывания в системе. Понятно, что вероятность перегрузки системы становится наибольшей именно тогда, когда по всем каналам прерывания начинают поступать с максимальной частотой. Это обстоятельство позволяет перейти от рассмотрения прерываний от всевозможных источников к рассмотрению прерываний от таймеров, генерирующих прерывания с частотой, соответствующей максимальной частоте прерываний от исходного источника. Именно так мы и будем поступать в дальнейшем. Необходимо отметить, что в практике встречаются задачи по обработке периодического прерывания таймера в реальном времени. Как правило, они связаны с опросом какого либо датчика и(или) управлением механической частью аппаратуры, например при интегрировании уравнений движения в системах телеуправления.

Сама по себе частота прерываний не может дать достаточного представления о том, способна система справляться с обработкой их или нет. Для того, чтобы характеризовать возможность системы справляться с решением периодической задачи, введем *коэффициент средней загрузки системы определенной задачей*, который определяется как отношение времени решения задачи к периоду повторения запросов на ее решение: $k = \frac{t}{T}$.

При наличии нескольких периодических задач возникает несколько коэффициентов средней загрузки по ним. Очевидно, необходимым условием решения таких задач совместно будет

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$$U = \sum_i k_i \le 1, \qquad \text{где}$$

k_i — коэффициент средней загрузки $i^{ой}$ задачей.

Величину *U* обычно называют коэффициентом использования процессора, или коэффициентом средней загрузки процессора.

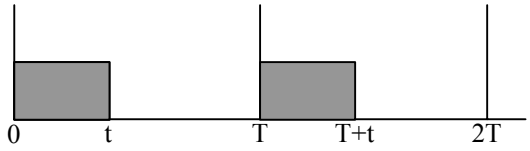

Рис. 1. Иллюстрация к определению коэффициента средней загрузки ЭВМ одной задачей

В случае, когда это условие выполняется, будем говорить, что *соблюдается временной баланс ЭВМ.* Следует подчеркнуть, что указанное соотношение является лишь необходимым условием бесконфликтного сосуществования нескольких задач. В зависимости от выбранной дисциплины планирования возможны пиковые перегрузки системы, которые и наблюдаются, если суммарная средняя загрузка *K* близка к единице.

Для сравнения характеристик различных дисциплин планирования необходимо сделать следующие предположения [1] о выполняемых задачах.

1. Запросы на выполнение каждой задачи поступают в систему периодически, с постоянным интервалом.
2. Каждая задача должна быть решена к приходу следующего запроса на ее решение.
3. Процессы решения разных задач независимы, запросы на решения одной задачи не зависят от состояния решения другой задачи.
4. Время обработки каждого запроса одной задачи постоянно и не меняется со временем. Под временем обработки понимается время, требуемое процессору для завершения решения без прерываний.
5. Выполнение задач допускает прерывание и вытеснение их более важными задачами (preemptive scheduling).

Все описываемые ниже дисциплины планирования объединяет общий принцип анализа выполнимости. От сложного и неудобного анализа множества временных характеристик отдельных задач они позволяют перейти к более простому анализу среднего использования ресурсов совокупностью этих задач. В том случае, если среднее использование ресурсов не превосходит некоторой фиксированной доли от общей производительности системы, дисциплина планирования обеспечивает решение к сроку всех задач. Такая особенность планирования существенно упрощает модификацию программ, достаточно лишь проверять, что использование ресурсов не превосходит единицы.

2. Дисциплина статических приоритетов

Во многих распространенных операционных системах реального времени для задач реального времени применяется дисциплина со статическими

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

приоритетами. Её применение можно встретить в таких системах как UNIX, QNX, VxWorks, Windows NT, LynxOS и многих других.

Суть этой дисциплины заключается в следующем. Имеется N уровней приоритета. Каждой задаче назначается постоянный приоритет. При одновременном наличии нескольких задач, готовых к выполнению на разных уровнях приоритета, всегда обрабатывается первой задача, имеющая более высокий приоритет. Самый высокий приоритет обычно присваивается задачам, имеющим наименьший период повторения. Можно показать, что такое правило является оптимальным, в том смысле, что не существует такой статической комбинации приоритетов, которая смогла бы обеспечить решение задач, если они не могут быть решены к срокам при назначении приоритетов по этому правилу [1].

Однако, даже следуя такому правилу распределения задач в системе не всегда возможно добиться отсутствия ***пиковых перегрузок***, если суммарная средняя загрузка близка к единице. Рассмотрим пример. Пусть имеется всего две задачи: t_1=1, T_1=2; t_2=1,5, T_2=3. $U = \frac{1}{2} + \frac{1.5}{3} = 1$, временной баланс соблюдается. Из рисунка 2 видно, что в системе возникает пиковая перегрузка, вторая задача не укладывается в срок.

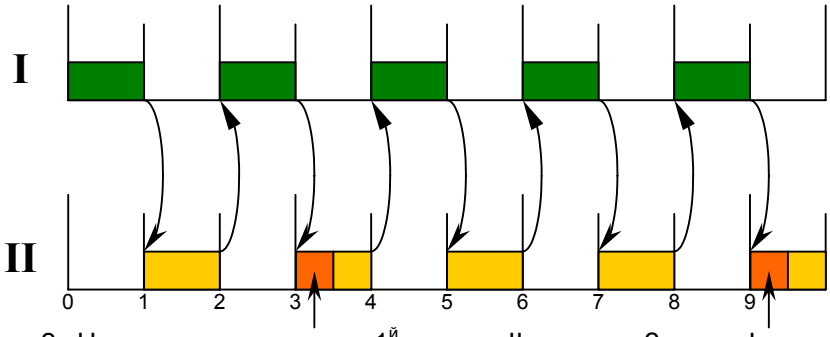

Рис. 2. Невыполненная часть 1й заявки II задачи. Задача I имеет больший приоритет, поскольку ее период наименьший. Поэтому по приходе второй заявки на её выполнение прекращается выполнение задачи II, что ведет к сбою задачи II

Пример показывает важность дисциплины планирования для отсутствия пиковых перегрузок в системе. Если время работы первой задачи снизить с 1 до 0,75, то перегрузка исчезнет, а суммарная средняя загрузка будет равна 87,5%.

Несмотря на такой видимый недостаток, система со статическими приоритетами очень проста в реализации и вследствие этого имеет весьма широкое распространение. В случае пиковых перегрузок дисциплина обеспечивает решение самых приоритетных задач в первую очередь.

В работе [1] было показано, что для n задач существует верхняя граница для коэффициента использования процессора, выше которой в системе появляются пиковые перегрузки:

$$U(n) = n(2^{1/n} - 1).$$

Следуя данной формуле, нетрудно убедиться, что при большом количестве задач порог загрузки процессора, гарантирующий отсутствие пиковых перегрузок, опускается до величины ***ln 2 — 69%***. Однако чем больше задач в системе, тем менее вероятно реализуется самый худший случай. Более детальный анализ [2] показывает, что средняя величина для такого порога составляет 88%.

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Учитывая современное развитие вычислительной техники, когда на рынке встречаются процессоры с одинаковой системой команд и производительностью, отличающейся на порядки, можно утверждать, что во многих конкретных случаях замена аппаратной части будет самым дешевым способом к достижению результата.

Тем не менее приходится констатировать, что часть производительности системы (31%) вообще не будет задействована, если не найдётся сторонних задач. Она будет востребована лишь в критических ситуациях пиковых перегрузок.

3. Дисциплина разделения времени

Как было показано в предыдущем параграфе, в системе со статически назначенными приоритетами возможны такие сочетания задач, которые приводят к пиковым перегрузкам системы. Дисциплина разделения времени является одним из подходов к решению проблемы пиковых перегрузок, который часто встречается на практике.

Идея такого подхода заключается в следующем. Рассмотрим те же самые две задачи, описанные в пункте 2. Разобьем решение задач на маленькие этапы, по 0,5 периода. Из рисунка 3 видно, что задачи стали укладываться в отведенные им времена.

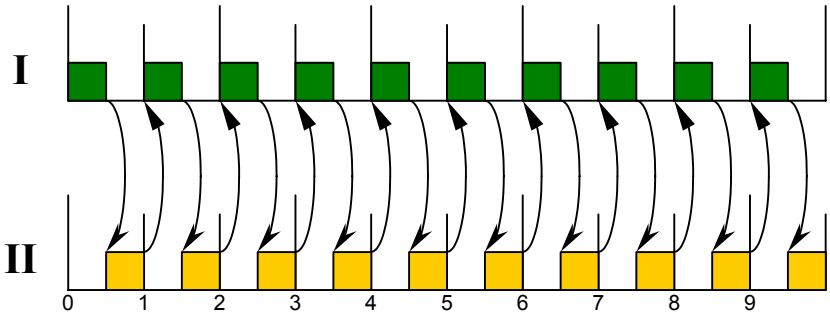

Рис. 3. При разделении процессорного времени на кванты по половине периода задачи начинают успевать к сроку

Обобщая этот подход на произвольные сочетания периодов повторения задач, можно пытаться выбрать некоторый квант времени, по истечении которого просто по очереди переключать задачи, которые еще не решены к сроку. Тогда погрешность в соблюдении временного баланса будет не больше величины кванта времени. Относительная же погрешность будет не больше величины кванта, деленной на самый маленький период повторения задач. Если устремить размер кванта к нулю, то можно избежать пиковых перегрузок при загрузке процессора U, сколь угодно близкой к 1. Кроме того, если знать соотношение времен выполнения задач, то обеспечить отсутствие перегрузок можно и при конечном кванте времени, как это видно из рисунка 3. Достаточно уже взять квант величины 0,5, и задачи будут успевать решаться без перегрузок в системе.

Математически эта дисциплина является попыткой распределить производительность одного реального процессора между несколькими виртуальными процессорами (каждой задаче по процессору), производительность которых составляет долю $k(i)$ от производительности

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

целой реальной ЭВМ. Решение одной задачи реального времени, как уже было отмечено, требует обеспечения достаточной производительности вычислительной машины. Суммируя эти условия, мы, очевидно, придем просто к необходимому условию, сформулированному в пункте 1.

Однако практически дисциплина равномерной загрузки наталкивается на трудности следующего содержания.

1. *Из-за ветвления программ никогда точно не известно время их выполнения, поэтому для гарантии отсутствия перегрузок квант времени желательно брать по возможности меньше.*
2. *Малость кванта времени служит причиной частых смен контекстов задач, что приводит к потере производительности.*
3. *Асинхронно возникающие переключения контекстов оставляют задачу, отдающую управление, в неизвестном состоянии, что может приводить к конфликтам с остальными задачами. Так, если одна задача займет некоторый общий ресурс и будет переключена на другую, то этим она блокирует возможность остальных задач работать с этим ресурсом. С учетом такого влияния задач друг на друга возможны пиковые перегрузки системы.*

В дополнение к этим трудностям необходимо отметить недостаток самого метода порождения виртуальных процессоров. Предположим, что имеется 10 одинаковых независимых задач со сроком 10, время вычисления которых равно 1. В случае, когда задачи решаются квазипараллельно, с небольшим квантом времени, отводимым каждой задаче, и разделением производительности процессора на 10 частей, результат каждой задачи будет реально готов к сроку, равному 10. Если те же задачи решать друг за другом, то результаты всех задач будут получены к сроку 10, однако реально результат первой будет готов к моменту времени 1, второй — к 2, и лишь самой последней — к 10. Таким образом, можно заключить, что разделение времени между независимыми задачами является примером “долгостроя”.

4. Дисциплина ближайшего срока

Своеобразным антиподом дисциплины разделения времени является дисциплина ближайшего срока (*Earliest Deadline First*), которая обладает минимумом переключений между задачами в процессе их решения. Принципы этой дисциплины весьма просты.

1. *Из всех задач, готовых к выполнению в данный момент времени, выбирается на обработку та, чей срок выполнения ближайший к текущему моменту времени. Таким образом, приоритеты между задачами расставляются динамически по принципу: ближнему сроку больший приоритет.*
2. *Приоритеты изменяются только при возникновении возмущений из внешней среды (прерываний), либо по завершении решения текущей задачи*

Строго говоря, формулировка дисциплины полностью задается первым принципом. Второй принцип указывает, что при реализации этой дисциплины достаточно иметь список задач, отсортированный по возрастанию сроков их выполнения. Изменения в этом списке происходят лишь под воздействием прерываний или по завершении выполняемой задачи.

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

В работе [1] эта дисциплина названа Deadline-driven scheduling. Там же приводится доказательство того, что дисциплина обеспечивает решение задач без пиковых перегрузок в системе вплоть до полной загрузки процессора. Однако это доказательство длинно и ненаглядно. В работах [3, 4] также рассмотрены возможные доказательства, основу которых составляет функция мгновенной загрузки, рассматриваемая ниже. Ниже приводится более простое и наглядное доказательство этого факта, придуманное автором.

Рассмотрим систему N задач реального времени в произвольный момент времени *t.* Обозначим через τ_i время, необходимое для окончания решения $i^{ой}$ задачи. Если задача к данному моменту уже решена и не ожидает решения, то положим $\tau_i=0$. Обозначим через T_i время, оставшееся задаче до срока ее решения.

Очевидно, что для отсутствия пиковых перегрузок необходимо, чтобы в любой момент времени выполнялись N условий:

$$\tau_i <= T_i, \quad i=1,..,N.$$

Введем в рассмотрение функцию мгновенной загрузки

$$F(t)=\sum_{i=1}^{N}\frac{\tau_i}{T_i}.$$

Функция F является почти всюду непрерывной, исключение составляют точки добавления новых задач.

Ясно, что если функция F<=1, то значит любое её слагаемое не больше единицы, и, следовательно, пиковых перегрузок нет. Кроме того, производная функции F по времени во всех точках, кроме точек разрыва:

$$F'=\lim\frac{\left(F(t+\Delta t)-F(t)\right)}{\Delta t}=\lim\frac{1}{\Delta t}\left(\left(\frac{\tau_1}{T_1-\Delta t}+...+\frac{\tau_j-\Delta t}{T_j-\Delta t}+...+\frac{\tau_N}{T_N-\Delta t}\right)-\sum_{i=1}^{N}\frac{\tau_i}{T_i}\right)=$$

$$=\lim\left(\frac{1}{\Delta t}\sum_{i=1}^{N}\left(\frac{\tau_i}{T_i-\Delta t}-\frac{\tau_i}{T_i}\right)-\frac{1}{T_j}\right)=\sum_{i=1}^{N}\frac{\tau_i}{T_i^2}-\frac{1}{T_j}\le\frac{1}{T_j}\left(\sum_{i=1}^{N}\frac{\tau_i}{T_i}-1\right)=\frac{1}{T_j}\left(F-1\right)\le 0,\ \Delta t\to 0.$$

При вычислении этой производной предполагалось, что в данный момент времени выполняется $j^{ая}$ задача. По определению дисциплины величина T_j является наименьшей из всех величин T. Так как F не больше единицы, то производная, очевидно, не больше нуля. Таким образом, приходим к заключению, что функция мгновенной загрузки при выбранной дисциплине не возрастает, если только ее начальное значение не превосходит единицу.

Дальнейшие рассуждения можно провести индукцией по числу задач. Для этого заметим, что в момент поступления нового запроса на решение задачи функция мгновенной загрузки F скачком вырастает на величину, равную средней загрузке системы этой задачей. Базой индукции служит тривиальное утверждение, что в отсутствие каких-либо задач система не перегружена. Индукционный переход состоит в том, что если функция F для k задач не превышает в любой момент времени средней суммарной загрузки по этим k задачам, то и для k+1 задачи функция F не превысит средней суммарной загрузки. Это с легкостью следует из того, что F не возрастает, стало быть, F не может превысить средней суммарной загрузки по k+1 задаче. Поскольку изначально мы предполагаем, что рассматриваемые N задач в среднем

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

систему не перегружают, то, значит, в любой момент времени F не превосходит единицы, то есть пиковых перегрузок нет***. Что и требовалось доказать.***

Приведенное доказательство хорошо объясняет ситуацию тогда, когда в систему не внесено никаких возмущений и изначально перегрузок не было. Действительно, доказательство факта невозрастания (убывания) F тесно связано с тем, что сама F не больше единицы. В противном случае F может не только не убывать, но будет обязательно обращаться в бесконечность, что будет соответствовать пиковой перегрузке системы. При этом последствия могут быть разные: либо не успевшая задача выполнится с опозданием, либо она вызовет эффект “домино” и повлечет перегрузки в остальных задачах. Во всяком случае, хотя бы одной перегрузки будет принципиально не избежать. В отличие от системы жестко заданных приоритетов, в такой ситуации трудно предсказать без специальных мер, решение какой задачи будет завершено позже срока. В этом отношении ее нельзя назвать стабильной.

Нетрудно видеть два интереснейших преимущества данной дисциплины перед остальными. Во-первых, число переключений контекстов решаемых задач минимально. Учитывая, что переключения тоже требуют времени, мы получаем наибольшую эффективность переключений. Во вторых, решение большинства задач идет “с упреждением”, то есть задача заканчивает решение за некоторое время до того, как наступит ее срок. Благодаря этому можно обеспечить устойчивое решение всех задач, так что возможные ветвления алгоритмов не будут приводить к перегрузкам.

Литература

[1] *Liu, C. L., and Layland J. W.* Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment // Journal of the ACM, Vol. 20, No. 1, January 1973. — pp. 46-61.

[2] *Lehoczky, J. P., Sha L., and Ding Y.* The Rate Monotonic Scheduling Algorithm – Characterization and Avarage Case Behaviour // Technical Report, Department of Statistics, Carnegie Mellon University, 1987.

[3] *Дашевский П. Г., Никольцев В. А.* Организация вычислительного процесса, управляющего несколькими объектами // Вопросы судостроения, серия XIII, вып. 2., 1973.

[4] *Дашевский П. Г., Никольцев В. А.* Организация обслуживания нескольких регулярных потоков заявок одной ЭВМ // Вопросы судостроения, вычислит. техника, вып. 5., 1974.

[5] *Шоу А.* Логическое проектирование операционных систем — М., Мир, 1981.

[6] *Klara Nahrstedt.* Process Management. // `http://www.courses.cs.uiuc.edu/~cs314/Lectures/MMOS/Process/talk/talk.html`