Процессы и потоки

Основные события, приводящие к созданию процессов.

- 1. Инициализация системы.
- Iniţializare sistemului.
- 2. Выполнение изданного работающим процессом системного запроса на создание процесса.
- Procesul activ prin mesaj de sistem cere crearea unui nou proces.
- 3. Запрос пользователя на создание процесса. Prin mesajul utilizatorului pentru crearea unui nou proces.
- 4. Инициирование пакетного задания.
- Iniţierea sarcinii.

Состояния процессов
Starea proceselor

- 1. Работающий (в этот конкретный момент использующий процессор).
- Procesul activ la moment.
- 2. Готовый к работе (процесс временно приостановлен, чтобы позволить выполняться другому процессу).

  Proces în stare de aşteptare a realizării altui proces.
- 3. Заблокированный (процесс не может быть запущен прежде, чем произойдет некое внешнее событие).
- Proces blocat. Este în stare de aşteptate a unui eveniment sau condiţie.

Necesitatea planificării proceselor.

1. La crearea unui nou proces se decide care proces de activat: procesul părinte sau derivat.

2. Planificare la finalizarea procesului. Se decide procesul care va fi activ. Dacă nu există procese în starea de așteptare se activează proces pustiu.

3. Planificare la blocarea procesului activ.

4. Planificare la apariţia întreruperii I/O. Se activează procesul care a foct blocat în așteptarea acestei unități de I/O.

Категории алгоритмов планирования
Tipurile algoritmilor de planificare

- 1. Системы пакетной обработки данных.

В системах пакетной обработки нет пользователей, сидящих за терминалами и ожидающих ответа. В таких системах приемлемы алгоритмы без переключений или с переключениями, но с большим временем, отводимым каждому процессу.

Sisteme de prelucrare a datelor în bloc.

- 2. Интерактивные системы.

В интерактивных системах необходимы алгоритмы планирования с переключениями, чтобы предотвратить захват процессора одним процессом. Даже если ни один процесс не захватывает процессор на неопределенно долгий срок намеренно, из-за ошибки в программе один процесс может заблокировать остальные. Для исключения подобных ситуаций используется планирование с переключениями.

Sisteme interactive

- 3. Системы реального времени.

В системах с ограничениями реального времени приоритетность, как это ни странно, не всегда обязательна, поскольку процессы знают, что их время ограничено, и быстро выполняют работу, а затем блокируются.

Sisteme de timp real.

Sarcinile algoritmilor de planificare

- **Pentru toare tipurile de siteme**

1. Repartizare cinstită — fiecare proces primește o porțiune de timp a procesorului (cuant de timp)
2. Realizarea forțată a cerințelor — control strict a cerințelor de realizare
3. Balansarea sistemei — încărcarea optimală a sistemului

Pentru sisteme de prelucrare a datelor în bloc.

1. Пропускная способность — un număr maximal de probleme
2. Оборотное время — minimizarea timpului de așteptare și prelucrare
3. Использование процессора — поддержка постоянной занятости процессора

Pentru sisteme interactive

1. Timpul de răspuns – răspuns imediat la mesaj
2. Соразмерность — выполнение пожеланий пользователя

- **Системы реального времени**

1. Окончание работы к сроку — предотвращение потери данных
2. Предсказуемость — предотвращение деградации качества в мультимедийных системах

Планирование в системах пакетной обработки данных
1. «Первым пришел — первым обслужен»

- Алгоритм без переключений «первым пришел — первым обслужен» является самым простым из алгоритмов планирования. Процессам предоставляется доступ к процессору в том порядке, в котором они его запрашивают. Чаще всего формируется единая очередь ждущих процессов. Как только появляется первая задача, она немедленно запускается и работает столько, сколько необходимо. Остальные задачи ставятся в конец очереди. Когда текущий процесс блокируется, запускается следующий в очереди, а когда блокировка снимается, процесс попадает в конец очереди.

- Основным преимуществом этого алгоритма является то, что его легко понять и столь же легко программировать.
- Недостаток - очень замедлят работу процесса, ограниченного возможностями процессора.

2. «Кратчайшая задача — первая»

- Если в очереди есть несколько одинаково важных задач, планировщик выбирает первой самую короткую задачу.

Это алгоритм без переключений для систем пакетной обработки, предполагающий, что временные отрезки работы известны заранее.

Алгоритм работает лишь в случае одновременного наличия задач.

3. Наименьшее оставшееся время выполнения

В соответствии с этим алгоритмом планировщик каждый раз выбирает процесс с наименьшим оставшимся временем выполнения. В этом случае также необходимо заранее знать время выполнения задач. Когда поступает новая задача, ее полное время выполнения сравнивается с оставшимся временем выполнения текущей задачи. Если время выполнения новой задачи меньше, текущий процесс приостанавливается и управление передается новой задаче. Эта схема позволяет быстро обслуживать короткие запросы.

4. Трехуровневое планирование

1. Впускной планировщик выбирает задание и передает его системе. Остальные задания остаются в очереди. Характерный алгоритм входного контроля может заключаться в выборе смеси из процессов, ограниченных возможностями процессора, и процессов, ограниченных возможностями устройств ввода-вывода. Также возможен алгоритм, в котором устанавливается приоритет коротких задач перед длинными. Впускной планировщик волен придержать некоторые задания во входной очереди, а пропустить задание, поступившее позже остальных.

2. Планировщик памяти. Как только задание попало в систему, для него будет создан соответствующий процесс, и он может тут же вступить в борьбу за доступ к процессору. Возможна ситуация, когда процессов слишком много и они все в памяти не помещаются, тогда некоторые из них будут выгружены на диск. Второй уровень планирования определяет, какие процессы можно хранить в памяти, а какие — на диске. Количество процессов, одновременно находящихся в памяти, называется степенью многозадачности.

3. Третий уровень планирования отвечает за доступ процессов, находящихся в состоянии готовности, к процессору. Когда идет разговор о «планировщике», обычно имеется в виду именно планировщик процессора. Этим планировщиком используется любой подходящий к ситуации алгоритм, как с прерыванием, так и без.

Планирование в интерактивных системах

1. Циклическое планирование

Каждому процессу предоставляется некоторый интервал времени процессора, так называемый квант времени. Если к концу кванта времени процесс все еще работает, он прерывается, а управление передается другому процессу. Если процесс блокируется или прекращает работу раньше, переход управления происходит в этот момент. Планировщику нужно поддерживать список процессов в состоянии готовности. Когда процесс исчерпал свой лимит времени, он отправляется в конец списка.

Переключение с одного процесса на другой занимает некоторое время — необходимо сохранить и загрузить регистры и карты памяти, обновить таблицы и списки, сохранить и перезагрузить кэш памяти и т. п. Переключение процессов или переключение контекста, как его иногда называют, занимает 1 мс, включая переключение карт памяти, перезагрузку кэша и т. п. Пусть размер кванта установлен в 4 мс. В таком случае 20 % процессорного времени уйдет на администрирование — это слишком много.

2. Приоритетное планирование

Основная идея алгоритма: каждому процессу присваивается приоритет, и управление передается готовому к работе процессу с самым высоким приоритетом. Чтобы предотвратить бесконечную работу процессов с высоким приоритетом,

планировщик может уменьшать приоритет процесса с каждым тактом часов. Если в результате приоритет текущего процесса окажется ниже, чем приоритет следующего процесса, произойдет переключение. Возможно предоставление каждому процессу максимального отрезка времени работы. Как только время кончилось, управление передается следующему но приоритету процессу.

Часто бывает удобно сгруппировать процессы в классы по приоритетам и использовать приоритетное планирование среди классов, но циклическое планирование внутри каждого класса.

3. Cîteva rînduri

Esența algoritmului:. Clasei cu procese cu cea mai mare prioritate îi este alocată un cuant de timp clasa următoare - două cuanturi și așa mai departe. Când procesul a folosit tot timpul alocat pentru el, el este mutat la clasa următoare.
Pentru separarea proceselor în clase se utilizează diverși algoritmi.

Cele mai des sunt utilizate patru clasă de prioritate, numite:

- terminale,
- intrare și ieșire,
- un scurt cuant
- un cuant lung.

4. Cel mai scurt proces - următorul

Procesele interactive repetă de multe ori modelul ”comanda de așteptare, executarea comanzii ..." Dacă luăm în considerare performanțele fiecărui proces ca o sarcină separată, putem minimiza timpul mediu de raspuns, cel mai scurt proces începe prima sarcină. Singura problemă este de a intelege care dintre procese este cele mai scurt proces.

O metodă se bazează pe o evaluare a duratei procesului, pe baza comportamentului anterior al procesului. Începe procesul care are timpul estimat cel mai mic.

5. Planificarea garantată

- O altă abordare a planificării este de a oferi utilizatorii promisiune reale, și apoi realizarea lor. Acestă promisiune este ușor de formulat și ușor de realizat, dacă aveți un procesor și sunt N utilizatori, va fi prevăzut pentru un utilizator 1/ N putere procesor.

- Pentru a îndeplini această promisiune, sistemul trebuie să păstreze evidența distribuției procesorului între procesele de la începutul fiecărui proces. Sistemul calculează apoi cantitatea de resurse a procesorului la care un proces este eligibil.

6. Planificarea cu loterie

Algoritmul se bazează pe distribuția între procese a biletelor de loterie pentru accesul la diverse resurse, inclusiv procesorul. Atunci când planificatorul trebue să ia deciderea, este selectate aleator bilet de loterie, iar proprietarul acestuia are acces la resursă. În ceea ce privește accesul la procesor, "loterie" pote apărea mai frecvent.

Proceselor mai importante se pot da bilete suplimentare. Fiecare proces va primi acel procent de resurse aproximativ egal cu numărul de bilete de care dispune.

Planuficare cu loterie este caracterizat prin mai multe proprietăți interesante. De exemplu, dacă la creare procesul a obține câteva bilete, apoi la loterie următoare șansele sale de a câștiga sunt proporționale cu numărul de bilete. Cu alte cuvinte, programarea loterie este foarte receptivă.

Procesele dependente pot face schimb de bilete, dacă este necesar.

7. Planificarea echitabilă

Sistem de operare acordă atenție la stăpînul procesului înainte de planificare.

În acest model, fiecare utilizator primește o anumită cantitate a procesorului și programatorul selectează procesului conform acestui fapt.

Planificarea sistemelor de timp real

Sisteme de timp real sunt împărțite în sistem real cu timp stabilit, ceea ce înseamnă că sunt termene strânse pentru fiecare activitate și sistem flexibil în timp real, în care timp graficului de realizare pote fi încălcat, ceia ce este nedorit, dar posibil.

În ambele cazuri, programul se desfășoară pe mai multe etape, fiecare dintre care este previzibil. Aceste procese sunt cel mai adesea scurte și își finalizează activitatea în scurt timp. Atunci când apare un semnal extern, programatorul trebuie să asigure respectarea calendarului de realizare.

continuare

Evenimentele externe la care sistemul trebuie să răspundă, poate fi împărțită în:

-periodice (care apar la intervale regulate)

- non-periodice (apar imprevizibil).

Pot fi evenimente periodice multiple și sistemul trebuie să se ocupe de ele. În dependență de timpul necesar pentru a procesa fiecare dintre evenimente poate fi faptul că sistemul nu este în stare să proceseze toate evenimentele în timpul util.

Sisteme de timp real care satisfac aceastei condiții sunt numite sisteme planificabile.

continuare

Algoritmi de planificare pentru sisteme în timp real poate fi **statici și dinamici**.

În primul caz, toate deciziile de planificare sunt realizate in avans, chiar înainte de lansarea sistemului.

În al doilea caz deciziile de planificare sunt realizate pe parcursul realizării.

Programarea static se aplică numai în cazul în care există informații fiabile cu privire la activitatea care urmează să fie făcută, și calendarul, care ar trebui respectat. Programarea dinamică nu are nevoie de astfel de restricții.