Управување со меморија

Оперативни системи 2018
Аудиториски вежби

Задача 1

- Компјутерски систем има доволно RAM меморија за 4 програми. Овие програми половина од времето чекаат на I/O уреди (idle). Колкав дел од CPU времето е потрошено залудно?

- РЕШЕНИЕ

  ½ -> веројатност дека процесот е неактивен.

  $(½)^4 = 1/16$ -> веројатност дека сите 4 процеси нема да бидат активни => 1/16 од времето сите процеси ќе се неактивни.

Задача 2

- Доколку две задачи стартуваат истовремено, при што секоја задача бара 10 минути од процесорското време, да се одреди:
  - За колку време ќе заврши последната задача доколку работат секвенцијално?
  - За колку време ќе заврши последната задача доколку работат паралелно?
- Да се претпостави дека двете задачи 50% од времето чекаат на I/O операции.

Задача 2 – решение

- Секвенцијално:
  - втората задача ќе заврши 40 минути откако ќе почне првата.
  - 20 мин. (за првата) + 20 мин. (за втората)
- Паралелно:
  - искористеноста на CPU е $1 - 0.5^2 = 0.75$
  - секоја задача ќе добие 0.375 од 1 CPU минута.
  - за да ефективно се извршува 10 минути, треба да поминат 10 / 0.375 минути = 26.67 минути.
  - затоа што работат паралелно, ќе завршат за 26.67 минути.

Задача 3

- Еден систем кој има 1MB меморија врши збивање на меморијата во текот на секоја секунда.
- Ако му треба 0.5μs (микросекунди) да ископира 1 бајт, а просечно дупките во меморискиот блок се 30% од блокот, колкав дел од процесорското време се користи за збивање на податоците?

Задача 3 – решение

- Бидејќи просечно дупките се 30% од меморискиот блок, тоа значи дека 70% од меморијата се податоци што треба да се ископираат од едно место на друго:
  $0.7 * 1\ MB * 0.5\ \mu s/B = 0.7 * 1024\ B * 1024 * 0.5\ s/B * 10^{-6} =$
  $0.7 * 1048576\ B * 0.5\ s/B * 10^{-6} =$
  $0.3670016\ s$
- Значи секоја секунда системот троши 0.3670016 секунди за збивање на меморијата.
- Оттука следува дека системот троши до 36,7% од времето за збивање (во најлош случај).

Задача 4

- Компјутерски систем што работи со замена (swapping) на задачи (процеси, програми) врши отстранување на малите празни делови од меморијата (дупки) преку процес на нивно спојување (memory compaction). Доколку празните и полните делови во меморијата се случајно распоредени и доколку времето потребно за читање или запишување 32-битен мемориски збор изнесува 10 ns, приближно колку време би било потребно за спојување на празните делови од меморија со големина од 128 MB?
- Забелешка: Да се претпостави дека зборот 0 е дел од празна меморија, додека последниот збор во меморијата содржи валидни податоци.

Задача 4 – решение

- Според поставеноста на задачата, се бара најлошиот можен случај, односно прераспоредување (читање и запишување) на целата меморија: секој збор се чита и презапишува на друга локација
  - Читање 32b (4B) бара 10 ns => читање 1B бара 2,5 ns;
  - Запишување 32b (4B) бара 10 ns => запишување 1B бара 2,5 ns;
  - Читање и запишување (прераспределба) на 1B бара 5 ns;
- Ако за 1B треба 5 ns, колку бајти ќе се прераспределат за 1 sec:
  - X = (1B)/(5 ns) = 200.000.000 B/s (рата на прераспределба) => за копирање 128MB (2^{27} B, или 1.34×10^8 B) потребно е време:
    - time = (2^{27} B) / (200.000.000 B/s) = 0,671 s = 671 ms;
- Овој број е песимистички, затоа што може да се заштеди време, доколку првиот празен дел од меморијата со големина од k бајти се наоѓа при дното, тој не мора да се копира.

Задача 5

- Да се направи споредба меѓу меморијата потребна за чување информации за слободната меморија со користење на: (а) битмапа, и (б) поврзана листа. Меморијата има големина од 128 MB и е алоцирана во делови од n бајти.
  - За поврзаната листа да се претпостави дека меморијата е составена од последователни сегменти од податоци и празни делови, секој со големина од 64 KB.
  - Исто така да се претпостави дека секој јазол во поврзаната листа има потреба од 32-битна мемориска адреса, 16-битна должина и 16-битно поле што покажува на следниот јазол.
- Колку бајти мемориски простор се потребни за двата начини? Кој е подобар?

Memory Management with Bitmap and Linked List

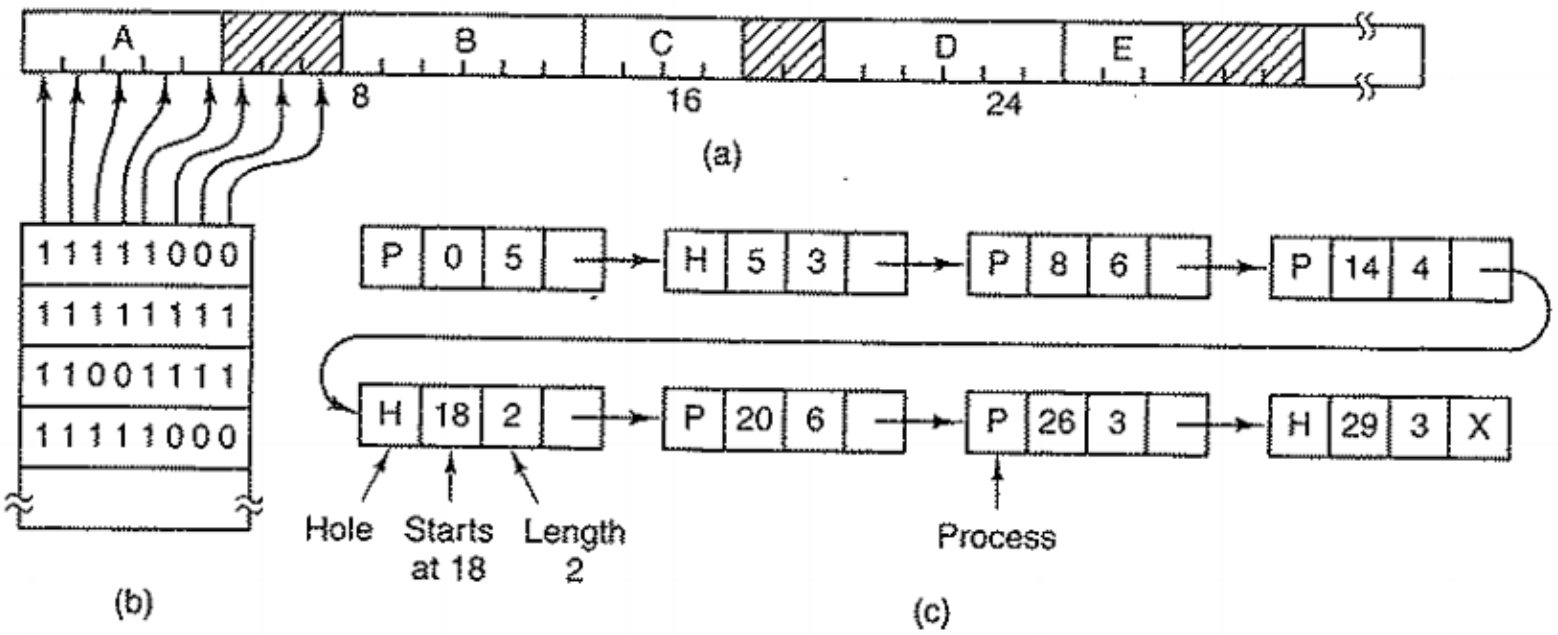


Figure 3-6. (a) A part of memory with five processes and three holes. The tick marks show the memory allocation units. The shaded regions (0 in the bitmap) are free. (b) The corresponding bitmap. (c) The same information as a list.

Задача 5 – решение

- Бит мапата има потреба од 1 бит по алокациска единица. Бидејќи има 2^{27} / n алокациски единици => тоа се 2^{27} / n бита => тоа се 2^{24} /n бајти.
- Поврзаната листа има $2^{27}/2^{16} = 2^{11}$ јазли, секој со по $8 = 2^3$ бајти за вкупно 2^{14} бајти.
- За мало n, поврзаната листа е подобра. За големо n, бит мапата е подобра.
- Пресечната точка може да се пресмета со изедначување на двете формули и решавање по n. Резултатот е 1 KB. За n помало од 1 KB, поврзаната листа е подобра. За n поголемо од 1 KB бит мапата е подобра.
- Инаку, секако претпоставката дека сегментите и празните места алтернираат на секои 64 KB е нереална.

Задача 6

- Еден систем има мемориски партиции со големини 100К, 500К, 200К, 300К и 600К. Да се изврши сместување на процеси со големини 212К, 417К, 112К и 426К, ако притоа се користат следните алгоритми за сместување во меморија:
  - First-fit
  - Best-fit
  - Worst-fit

- Со кој од овие алгоритми најефикасно се користи меморијата?

Задача 6 – решение

- Алгоритамот First-fit
  - За сместување на n бајти се користи првиот расположив слободен блок поголем од n;
  - Процесите се сместуваат во следниов редослед: 212К, 417К, 112К и 426К;

100	500			200	300	600	
100	212	112	176	200	300	417	183

  - Процесот со големина 426К нема кај да се смести.

Задача 6 – решение

- Алгоритамот Best-fit
  - За сместување на n бајти се користи најмалиот расположив слободен блок поголем од n;
  - Процесите се сместуваат во следниов редослед: 212К, 417К, 112К и 426К;

100	500		200		300		600	
100	417	83	112	88	212	88	426	174

  - Сите процеси се распределени.

Задача 6 – решение

- Алгоритамот Worst-fit
  - За сместување на n бајти се користи најголемиот расположив слободен блок поголем од n;
  - Процесите се сместуваат во следниов редослед: 212К, 417К, 112К и 426К;

100	500		200	300	600		
100	417	83	200	300	212	112	276

  - Процесот со големина 426К нема кај да се смести.

Задача 6 – решение

- Во овој пример најефикасно користење на меморијата има со Best-fit алгоритмот.
  - Сепак, во одредени случаи може да направи голема фрагментација – да имаме многу слободни парчиња меморија со мала големина, во која не може да се смести ниту една програма;
- Worst-fit алгоритмот е полош од First-fit алгоритмот, бидејќи прави поголема фрагментација на блоковите во меморијата.
  - Сепак, во одредени сценарија дозволува сместување на повеќе помали програми;

Задача 7

- Еден логички адресен простор има 32 страници, секоја со големина 1024 збора и се пресликува во физичка меморија со големина од 8 рамки.
- Од колку бита се состои:
  - логичката адреса;
  - физичката адреса;

Задача 7 – решение

- Логичка адреса: 15 бита
  - 5 за број на страница и 10 за offset
- Физичка адреса: 13 бита
  - 3 за број на рамки и 10 за offset

Задача 8

- За декадните виртуелни адреси: 20000, 32768, 60000, да се одреди бројот на виртуелната страница, и отстапувањето (offset) во рамките на истата, доколку страницата има големина од:

  а) 4KB б) 8KB

- РЕШЕНИЕ:

  a - address; **p** - page size; **pn** - page number; **off** - offset

  a = p * pn + off

  4KB = 4096B; page: 20000/4096 = 4 offset = 20000 – 4 * 4096

  8KB = 8192B; page: 20000/8192 = 2 offset = 20000 – 2 * 8192

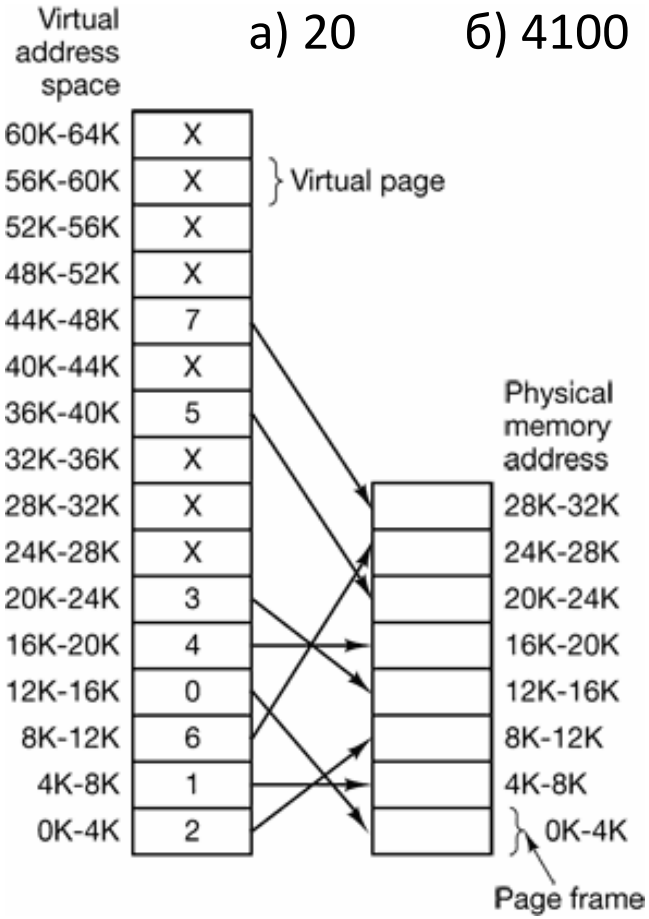
  Ако страната е со големина од 4KB, паровите (page, offset) се (4, 3616), (8, 0), и (14, 2656).

  Ако страната е со големина од 8KB паровите (page, offset) се (2, 3616), (4, 0) и (7, 2656).

Задача 9

- Дадена е табелата на страници. Да се одреди физичката адреса што одговара на секоја од дадените виртуелни адреси:

а) 20 б) 4100 в) 8300



РЕШЕНИЕ:

а) 20 е адреса од првата виртуелна страна. **Page = 0, offset = 20.** Се мапира во рамка 2, што почнува од адреса 8К => 8К=8192, физичка адреса = 8192 + 20 = 8212

б) 4100 е адреса од втората виртуелна страна. **Page = 1, offset = 4100 – 4096 = 4**. Се мапира во рамка 1, што почнува од адреса 4К => 4К=4096, физичка адреса = 4096 + 4 = 4100

в) 8300 е адреса од третата виртуелна страна. **Page = 2, offset = 8300 – 8192 = 108**. Се мапира во рамка 6, што почнува од адреса 24К => 24К = 24 * 1024 = 24576, физичка адреса = 24576 + 108 = 24684

Задача 10

- Големината на просторот на дискот што мора да биде овозможена за чување на страници е директно поврзана со максималниот број на процеси: n, бројот на бајти во виртуелниот адресен простор: v, и бројот на бајти во RAM: r. Дајте го изразот за големината на дискот во најлошото сценарио. Колку е тој израз реалистичен?

- РЕШЕНИЕ:
- Тоталниот виртуелен адресен простор потребен за сите процеси е n * v. Но, r бајти може да бидат во RAM. Па, така, формулата е: n * v – r.
- Овој израз не е многу реалистичен, поради тоа што ретко се случува сите n процеси да бидат истовремено активни, а и ретко некој од нив ќе го користи целиот виртуелен адресен простор.

Задача 11

- Извршувањето на една инструкција трае 10 ns, а при page fault се потребни дополнителни n ns. Да се прикаже формула за ефективното време на извршување на инструкциите доколку page fault се случува на секои k инструкции.

- РЕШЕНИЕ:

  Page fault се случува на секои k инструкции и додава дополнително време од n/k nsec во просек, па просечно за инструкција е потребно:

  - 10 + n /k nsec

Задача 12

- Еден компјутерски систем има 32 битен адресен простор и 8KB страница. Табелата на страници целосно се наоѓа во хардвер, при што секој запис се состои од еден 32-битен збор. Кога процесот ќе започне со работа, табелата на страници се копира во хардвер од меморија со брзина од еден збор на секои 100 ns. Ако секој процес работи по 100 ms (со вклучено време за снимање на табелата на страници), колкав процент од процесорското време е посветен на снимање на табелите на страници?

- РЕШЕНИЕ:

  Табелата на страници содржи $2^{32} / 2^{13}$ записи, односно 524.288 записи. За вчитување на табелата на страници е потребно: 524.288 * 100 ns = 52 ms. Ако еден процес добива 100 ms, тогаш 52 ms се трошат за вчитување на табелата на страници, а процесот работи 48 ms. Следи дека 52% од времето се троши на вчитување на табелите со страници.

Задача 13

- Компјутерски систем чии процеси имаат 1024 страници во адресниот простор, ги чува табелите на страници во меморија. Изгубеното време потребно да се прочита збор од табелата на страници е 5 ns. За зголемување на брзината компјутерот има TLB (Translation Lookaside Buffer) што содржи 32 парови (виртуелна страна – физичка рамка) и може да врши пребарување за 1 ns. Колкава рата на погодок е потребна за да се намали средното изгубено време на 2 ns?

- РЕШЕНИЕ:
- Ефективното време за една инструкција е 1*h + 5*(1 – h), каде h е ратата на погодок.
- Ако се изедначи оваа равенка со 2 и се реши по h, се добива дека h мора да биде барем 0.75.

Задача 14

- Нека w=7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 е низа од побарувања на страници (page reference stream).
- Нека има 3 рамки во физичката меморија и нека во почетокот тие се празни.
- Колку грешки при страничење ќе бидат генерирани ако како алгоритам за замена се користи:
  - FIFO (First in First out)
  - оптималниот алгоритам на Belady
  - LRU (Least Recently Used)
  - LFU (Least Frequently Used)

Задача 14 – решение

- Наједноставниот алгоритам за замена на страници е FIFO алгоритамот. Овој алгоритам ја поврзува секоја страница со времето кога оваа страница била донесена во меморија. Кога треба да се замени страница, се избира онаа која што најдолго не била заменета.
  - Овој алгоритам се имплементира со помош на редица, во која се чуваат сите страници во меморијата. Се заменува страницата која што се наоѓа на главата на редот, а потоа кога ќе се донесе новата страница таа се вметнува на крај на оваа редица

Задача 14 – решение

Рамка	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	7*	7	7	2*	2	2	2	4*	4	4	0*	0	0	0	0	0	0	7*	7	7
1		0*	0	0	0	3*	3	3	2*	2	2	2	2	1*	1	1	1	1	0*	0
2			1*	1	1	1	0*	0	0	3*	3	3	3	3	2*	2	2	2	2	1*
	*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

- Со примена на FIFO алгоритамот, детектирани се 15 грешки

Задача 14 – решение

- Алгоритмот на Belady претпоставува дека однапред се знае низата од побарувања на страници
- Во пракса тоа не е случај, но овој алгоритам се користи за оценување на перформансите на другите алгоритми
- Со овој алгоритам се врши заменување на онаа страница која најдоцна ќе биде повторно побарана

Задача 14 – решение

Рамка	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	7*	7	7	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	7*	7	7
1		0*	0	0	0	0	0	4*	4	4	0*	0	0	0	0	0	0	0	0	0
2			1*	1	1	3*	3	3	3	3	3	3	3	1*	1	1	1	1	1	1
	*	*	*	*		*		*			*			*				*		

- Со примена на оптималниот алгоритам за замена на страници, детектирани се 9 грешки

Задача 14 – решение

- LRU (Least Recently Used) алгоритамот ја поврзува секоја страница со времето кога оваа страница последно била употребена.
- Кога страница мора да биде заменета, LRU ја избира страницата која што не била употребена подолг временски период
  - Овој алгоритам наликува на оптималниот, само со таа разлика што времето не се гледа во иднина, туку во минатото

Задача 14 – решение

Рамка	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	7*	7	7	2*	2	2	2	4*	4	4	0*	0	0	1*	1	1	1	1	1	1
1		0*	0	0	0	0	0	0	0	3*	3	3	3	3	3	0*	0	0	0	0
2			1*	1	1	3*	3	3	2*	2	2	2	2	2	2	2	2	7*	7	7
	*	*	*	*		*		*	*	*	*			*		*		*		

- Со примена на LRU алгоритамот за замена на страници, детектирани се 12 грешки

Задача 14 – решение

- LFU (Least Frequently Used) ја исфрла онаа страница која била најмалку користена
  - Се користи бројач за секоја страница, со бројот на референцирања (пристапи)
- Ако има повеќе страници со ист број на пристапи, може да се употреби некој друг алгоритам за избор меѓу нив
- Во следниот пример се користи случаен избор помеѓу страниците со иста фреквенција на побарување

Задача 14 – решение

Рамка	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	7*	7	7	7	7	3*	3	4*	4	3*	3	3	3	3	3	3	1*	7*	7	1*
1		0*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2			1*	2*	2	2	2	2	2	2	2	2	2	1*	2*	2	2	2	2	2
	*	*	*	*		*		*		*				*	*		*	*		*

- Со примена на LFU алгоритамот за замена на страници, детектирани се 12 грешки

Задача 15

- Компјутерски систем располага за секој процес со 65536 бајти адресен простор поделен во страници од по 4096 бајти. Одредена програма со големина 32768 бајти, има податочен дел со големина 16386 бајти и стек сегмент со големина 15870 бајти. Дали оваа програма ја собира во адресниот простор? Доколку страниците се со големина од 512 бајти, дали ќе се променеше нешто? Страница не може да содржи делови од различни сегменти.

- РЕШЕНИЕ:

  Програмата е 32768/4096 = 8 страници, податоците се 16386/4096 = 5 страници и стекот е 15870/4096 = 4. Програмата не ја собира поради тоа што потребни се 8 + 5 + 4 = 17 страници со големина 4096 бајти, а имаме 65536/4096 = 16 страници.

  Додека, пак, доколку страниците беа големи 512 бајти: Програмата е 64 страници, податоците се 33 страници и стекот 31, вкупно 128 страници со големина 512 бајти. Вкупниот адресен простор исто така располага со 65536 / 512 = 128 страници.

Задача 16

- Заклучено е дека бројот на извршени инструкции помеѓу page faults е пропорционален со бројот на рамки алоцирани за соодветната програма. Доколку меморијата се зголеми 2 пати, средниот интервал помеѓу page faults е исто така двојно поголем. Претпоставете дека нормална инструкција се извршува за 1 микросекунда, но доколку се појави page fault потребни се 2001 микросекунди. Доколку за да се изврши една програма потребно е време од 60 секунди, во кое што се јавуваат 15000 page faults, колку време ќе биде потребно да се изврши истата програма доколку се удвои расположливата меморија?

Задача 16 – решение

- 15000 page faults * 2000 микросекунди = 30 секунди overhead. Ова значи дека половина од времето се троши на page faults, а половина на извршување на самата програма.
- Доколку се удвои меморијата, бројот на page faults би се намалил на 7500, па overhead–от би се намалил на 15 секунди, така што, програмата активно би се извршувала за 45 секунди.

Прашања?