

POWERLEVELING GUIDE ZA 6KA PO OS (VTOR KOLOKVIUM **TEORIJA** EDITION)



мацо



Wake up forsaken, време е да метарејмаме за оценка!

1. Управување со меморија (1 - 20|v|)

1)

Question 1
Incomplete answer
Marked out of 31.00
1 Flag question

Еден компјутерски систем има 256 KB виртуелна меморија. Големината на физичката меморија е 8 KB, организирана во рамки со големина од 2048 B. Времето за вчитување на податок при погодок е 300 наносекунди, а времето потребно за добивање на податок при промашуваче е 20 микросекунди.

One computer system has 256 KB virtual memory. The physical memory is 8 KB, organized in frames of size 2048 B. To read data when hit the system needs 300 nanoseconds, whereas to read data when miss happens the system needs 20 microseconds.

Форматот на адресите на овој систем е / System address format:

Virtual address		Physical address	
Page	Offset	Frame	Offset
2^7	2^11	2^2	2^11

Нека процесот генерира адреси кои пристапат на страниците 7d, 1d, 8, 7d, 63, 8, 5, 3c, 63, 8, 63, 5 и алгоритмот за замена е First In First Out. Пополнете ја следната табела, каде во колоната Погодоци треба да наведете 1 или 0 во зависност од тоа дали има погодок или не, а во колоната Рамка треба да ја наведете рамката во која е сместена соодветната страница.

The process generates addresses that belong to the pages 7d, 1d, 8, 7d, 63, 8, 5, 3c, 63, 8, 63, 5 and the algorithm for replacement is First In First Out. Fill the following table by entering 1 or 0 in the Hits column to show whether a hit or miss happened, and enter the frame for that page in the Frame column.

Страница / Page (hex)	Погодоци / Hits (0/1)	Рамка / Frame
7d	0	0
1d	0	1
8	0	2
7d	1	0
63	0	3
8	1	2
5	0	0
3c	0	1
63	1	3
8	1	2
63	1	3
5	1	0

Вкупното време на извршување на мемориите барања на оваа програма е / Total execution time for this program is 1150 ns.

Виртуелната адреса / Virtual address 28F9 (hex) се мапира во физичката адреса / is mapped into physical address F9 (hex).

Физичката адреса / Physical address 588 (hex) се мапира во виртуелната адреса / is mapped into virtual address 2D88 (hex).

1. Претварање на вредности:

Виртуелна меморија: 256KB = $256 * 1024 = 262\,144$ B
Физичка меморија: 8KB = $8 * 1024 = 8192$ B
Организирани во рамки: 2048B
Време на погодок: 300ns
Време на промашок: 20μs = $20 * 1000 = 20\,000$ ns

2.

Форматот на адресите на овој систем е / System address format:

Virtual address		Physical address	
Page	Offset	Frame	Offset
2^7	2^11	2^2	2^11

Табела за формат на адреси:

Офсет се пресметува како 2 на степен од големината рамките и се запишува степенот.

2048B = $2^{11} = 11$ офсет = овој степен се запишува во двата офсет полиња.

Страница полето се пресметува како виртуелната меморија поделена со големината на рамките и се зима 2 на степен од резултатот.

$262\,144 / 2048B = 128 = 2^7$ = овој степен се запишува во полето за страница.

Рамка полето се пресметува на ист начин како страница полето само сега физичката меморија е поделена со големината на рамките.

$8192B / 2048B = 4 = 2^2$ = овој степен се запишува во полето за рамка

3.

Страница / Page (hex)	Погодоци / Hits (0/1)	Рамка / Frame
7d	0	0
1d	0	1
8	0	2
7d	1	0
63	0	3
8	1	2
5	0	0
3c	0	1
63	1	3
8	1	2
63	1	3
5	1	0

Табела за генерирање на адреси:

БИТНО!

Алгоритам: FIFO (First In First Out)

Погодок: 1

Промашок: 0

Рамки: 4 -> 2^2 од претходната табела, се нула индексират (0 - 3)

7d: 0 = Бидејќи прв пат се генерира, се сместува во рамка 0

1d: 0, се сместува во рамка 1

8: 0, се сместува во рамка 2

7d: 1 = Погодок! Не се прв пат генерира, смести го во рамката што беше прв пат генериран. Во овој случај 7d беше сместен во рамка 0.

63: 0, смести го во рамка 3

8: 1 = Погодок! Не се прв пат генерира, смести го во рамката што беше прв пат генериран. Во вој случај 8 беше сместен во рамка 2.

5: 0, смести го во рамка 0 = Почнува пак од рамка 0 бидејќи имаме само 4 рамки на располагање (0 - 3)

3c: 0, смести го во рамка 1

63: 1, 3

8: 1, 2

63: 1, 3

5: 1, 0

Третата колона е помошна (не се оценува)

4.

Вкупното време на извршување на мемориите барања на оваа програма е / Total execution time for this program is 1150 ns.

Формула: БрНаПогодок * ВремеНаПогодок + БрНаПромашок * ВремеНаПромашок

Време на погодок: 300ns

Време на промашок: 20μs = $20 * 1000 = 20\,000$ ns

Број на Погодок: 6 = Ги броиме 1 во претходната табела во Погодоци колоната

Број на Промашок: 6 = Ги броиме 0 во претходната табела во Погодоци колоната

Вкупно време = $6 * 300ns + 6 * 20\,000ns = 1800ns + 120\,000ns = 121\,800ns$

5.

Виртуелната адреса / Virtual address 28F9 (hex) се мапира во физичката адреса / is mapped into physical address F9 (hex).
Физичката адреса / Physical address 588 (hex) се мапира во виртуелната адреса / is mapped into virtual address 2D88 (hex).

Мапирање виртуелна во физичка:

Виртуелната адреса: 28F9 (hex) = 0010 1000 1111 1001 (binary)

Ги броим битовите во бинарната адреса = 16 бита

Треба да се 18 бита, фалат уште 2.

18те бита ги одредуваме со првата табела од задачата. Ги собираме степените во Виртуелната адреса колона = $2^7 + 2^{11} = 18$.

Бидејќи ни фалат уште 2 лепеме 0 на почетокот од бинарниот број.

Модифицираната бинарна адреса = 00 0010 1000 1111 1001

Следно ја делеш адресата на два дела. Првиот дел ќе е 7 бита, а вториот дел ќе е 11 бита.

Поделбата ја одредуваме според степените во првата табела 2^7 и 2^{11} .

Поделената адреса: 0000101 | 0001111001

Првиот дел со 7 бита во декадно е 5.

Во втората табела тоест таа што ни е за генерирање на адреси 5 има рамка 0 (гледаш од долу па нагоре).

Вредноста на првиот дел го менеш со 00 (2 бита за големината на рамката на физичката адреса).

00 | 0001111001 -> 00001111001 -> F9 = Го претварааш назад во hex.

Мапирање физичка во виртуелна (исто како претходно само обратно)

Физичка адреса: 588 (hex) = 0101 1000 1000 (binary)

Ги броим битовите во бинарната адреса = 12 бита

Треба да се 13 бита, фале уште 1.

13те бита ги одредуваме со првата табела од задачата. Ги собираме степените во Физичката адреса колона = $2^2 + 2^{11} = 13$.

Бидејќи ни фале уште 1 лепеме 0 на почетокот од бинарниот број.

Модифицираната бинарна адреса = 0 0101 1000 1000

Следно ја делеш адресата на два дела. Првиот дел ќе е 2 бита, а вториот дел ќе е 11 бита.

Поделбата ја одредуваме според степените во првата табела 2^2 и 2^{11} .

Поделената адреса: 00 | 01010001000

Првиот дел со 2 бита во декадно е 0.

БИТНО! Сега гледаш кај колоната за рамки а не кај страниците!!!

Во втората табела тоест таа што ни е за генерирање на адреси 0 има страница 5 (гледаш од долу па нагоре).

Вредноста на првиот дел го менеш со 00101. (Уште една 0 ставеш на почетокот за да се совпадне големината)

00101 | 01010001000 -> 00101010001000 -> 2D88 = Го претварааш назад во hex

Нова компютарски систем има страници со големина од 512 Б, виртуелен адресен простор од 16 КБ и физичка меморија со големина 4 КБ. Притоа, дадена програма ја има следната табела со страници (припокажува се само валидните во експлицитната форма):

Page (hex)	Frame (hex)
0	3
1	0
2	0
3	2
4	6
5	4
7	1

Системот користи Translation Lookaside Buffer со капацитет од страници 0, 3 и 7 со време за пристап од 10 наносекунди, времето за пристап до главната меморија е 200 наносекунди, а времето за носење на страна од дискот е 2 микросекунди. Физичките адреси на овој систем се:

Виртуелна адреса	Физичка адреса		
Page	Offset	Frame	Offset
5	9	3	9

На почетокот на виртуелната адреса доаѓаат 5 бита за страници бидејќи $2^5 = 32$, што е бројот на страници од 512B што ги бери во В.А.П. од 16KB. Се смета дека се доаѓаат 3 бита за рамка кај физичката адреса бидејќи $8 \times 128 = 1024$. Offset-от е 9 бита што 7 бита + 2 бита.

Табелата на страници за свој процес има вкупно 32 записи, а во физичката меморија има вкупно 8 страници.

Одговорете ги следните прашања доколку имате барање за адресата 635 (hex):

011000110101 бинарно. Последните 9 бита се offset, а останатите бидејќи 3 бита на страници 3 во hex. Тоа се наоѓа во TLB и во табелата со страници на ова подрачје и во датите. До кога се пристапува во евраќи, и до датите нема потреба да се пристапува.

• Дали имате попогод во TLB табелата? **да** ✓

• Дали имате попогод во табелата со страници? **да** ✓

• До TLB табелата се пристапува 1 пат(и), до главната меморија 1 пат(и) и до дискот се пристапува 0 пат(и).

• За колку време ќе се добие доаѓаањето од адресата 635 (hex)? 210 н наносекунди и до главната меморија

• Физичката адреса која ќе се побара ќе биде 435 (hex) (Ако не може да се определи, наведете /) **и/или на времето на пристап до TLB**

Offset-от се пресметува, но 011 се заменува со 010 бидејќи на страници 3 соодветната рамка 2. Преглеано во hex конечниот број е 435.

Дополнително, оваа оперативност систем користи доколку се замина на нивната меморијска страница со адресирање JRU „Jailing“, каде за физичката адреса се чуваат пресметките од последните 3 интервала (clock tick) . Нива се референцираат следните страници во секое од интервалите (интервал 5 е последни):

Интервал	Пристапени страници (clock tick)	(hex)
1	1, 3, 6, 7	
2	1, 7	
3	1, 6, 7	
4	0, 3, 6	
5	0, 3, 7	

Во овој оперативен систем, за секое страници ќе се чуваат 3 битови за одредување на пристапите во минатите интервали.

Пополнете ги 8 битовите на својот од страниците, по заземањето на интервал 5:

Page (hex)	Битови
0	110
1	001
3	110
8	011
7	101

Секое бит претставува дали страницата била референцирана во интервал 5, 4 или 3 соодветно.

Ако физичката меморија е полна ќе се замени страницата: 1 бидејќи најдолго време не била референцирана

За овој прашање не е задолжително објаснувањето. Ова поле ќе се гледа само на увид.

Give your resons

Виртуелна адреса		Физичка адреса	
Page	Offset	Frame	Offset
5	9	3	9

Ако физичката меморија е полна ќе се замени страницата 1 бидејќи најдолго време била референцирана.

Еден систем користи страничење со инвертирана табела чија содржина е следната:

pid	page
1	4
2	4
1	3
5	2
1	9
2	5
1	6

Според табелата, може да се заклучи дека бројот на виртуелни страници е (/ ако не може да се определи), а бројот на рамки е (/ ако не може да се определи). Кај овој тип на страничење, големината на табелата на страници зависи од .

Ако процесот со PID=1 генерирал виртуелна адреса која припаѓа на страница 3, тогаш, според дадената табела, страницата се наоѓа во рамка (/ ако не може да се определи).

7. Бројот на виртуелни страници: /
 Бројот на рамки: 7 -> броеш колку редици од страници има во табелата.
 Големината на табелата зависи од бројот на рамки -> колку рамки има толку редици има
 Страницата се наоѓа во рамка 2 (рамките се нула индексирани) -> насочи се кон црвениот маркер на табелата

4)

Дадена е шема во која з-ознака дека меморијскиот блок е веќе исполнет со некој процес, а - ознака дека меморијскиот блок е слободен за алокација:

$[X] \mid [-] \mid [-] \mid [X] \mid X \mid [-] \mid [X] \mid [-] \mid [-] \mid [X]$

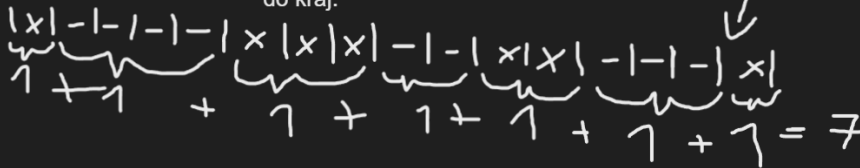
Доколку се користи битпазлата, тогаш најдобрата вредност за блокот од шемата ќе биде , а доколку се користи поврзаната листа, таа ќе биде составена од делни и содржината на првите три делни ќе биде: (дозволените се нумерички вредности, буквите P или H на латиница, значи да бидат одредени со едно празно место, пр. H 1 12 P 11 3).

При пребарување на слободни блокови во меморијата, според перформанси дава .

Ако е потрпено да се резервираат два блока, тогаш подобро BEST FIT ќе се резервира блокот со индекс . Според FIRST FIT ќе се резервира блокот со индекс . Ако се користи NEXT FIT, под претпоставка дека претходно бил доделен блок со адреса 10, ќе се резервира блокот со индекс . Ако се користи QUICK FIT, тогаш се користи .

1. 0 1111 000 11 00 111 0 | 'x' = 0, '-' = 1, za sekoj 'x' i za sekoj '0' gremo na 0 i 1 u skladu s tabelom.

2. 7 JAZLI | Broish gi 'x' i broish '- ', prviot 'x' e 1iot jazol, a narednite 4etiri '- ' se 2riot jazol itn do kraj.



3. P = proces = 'x', H = hole = '-'
treba da se napisati prвите 3 jazli -> ova e bitno, citaj ubavo pred da gi pisuvas
P 0 1 = prvot jazol x e proces, pocnuva kaj indeks 0 i ima dolzina 1
H 1 4 = posle imame jazol od '-' toa e hole, pocnuva kaj indeks 1 i ima dolzina od 4
P 5 3 = tretiot jazol x e proces, pocnuva kaj indeks 5 i ima dolzina od 3
kraen rezultat: P 0 1 H 1 4 P 5 3 -> site gi razdvojuvas so prazno mesto

4. Linked List

5. Bitno: Sekoj jazol so ' - ' oznacuja jazol so slobodni memoriski blokovi, a sekoj poseben ' - ' e poseben sloboden memoriski blok

Treba da rezerviras 2 memoriski bloka.

BEST FIT traze najdobriot prv jazol so se sovpagja | pocnuvas od indeks 0 toa e prviot jazol x. Iteriras niz jazlite se dodeka ne dojdes do indeks 8. Tuka ima dva ' - ' so e točna brojka na memoriski bloka so ni treba da rezervirame

FIRST FIT traze prvot jazol kade mozes da gi smestes memoriskite bloka | pocnuvas od 0 toa e prvio jazol x. Iteriras niz jazlitse se dodeka ne dojdos do indeks 1. Tuka ima 4etiri ' - ' so e prvio jazol so dovolno slobodni mesta za da se smestat memoriskite bloka

NEXT FIT cuva istorija od poslednata rezervacija vo ovoj slucaj poslednata bila na indeks 10, i trazes first fit posle taj indeks toest indeks 12

QUICK FIT mora da ima tabela kade sekoja stavka pokazuje kon lista od slobodni blokovi so ista golemina

5)

Question 4

Correct

Marks: 8.33 out of 8.33

Flag question

⚙ Edit question

Еден компјутерски систем имплементира виртуелна меморија со големина на страница од 16 бајти. Капацитетот на виртуелниот простор изнесува 512 бајти. Капацитетот на физичката меморија е 128 бајти. Содржината на **дел** од табелата на страниците е следна:

Индекс	Рамка	Валидност
0	2	1
1	3	1
2	4	0
3	1	1
4	5	1
5	1	0
6	7	1

Големиот на виртуелната адреса е бита, од кои бита се за страница, а бита се за поместување.

Големиот на физичката адреса е бита, од кои бита се за рамка, а бита се за поместување.

Целата табелата на страниците има вкупно ставки.

Виртуелната адреса **2010010111** се наоѓа во физичка адреса (битови без празни места, внесете / ако не може да се определи).

1. Големината на виртуелната адреса: $512B = 2^9$
Бита за страница: $512B / 16B = 32 = 2^5$
Бита за поместување: $9 - 5 = 4$

Големината на физичката адреса: $128\text{B} = 2^7$
 Бита за рамка: $128\text{B} / 16\text{B} = 8 = 2^3$
 Бита за поместување = $7 - 3 = 4$

Целата табела на страници
има вкупно ставки: $512B / 16B = 32$

001000111 ја поделиш на два дела, првиот дел ќе е 5 бита, а вториот 4

00100 | 0111 -> 00100 -> 4 -> го тражеш во
индекс колоната 4 се мапира за 5 -> 00100 =
0101

0101 | 0111 -> 01010111 -> 57(hex)

6)

Question 1
Incorrect
Mark 0.00 out of 4.00
Flag question

Виртуелната адреса на еден систем има 5 бита за првото ниво, 5 бита за второто ниво и 6 бита за офсет.

Првото ниво ќе се состои од табела/и со по ставки.

Второто ниво ќе се состои од табела/и со по ставки. Секоја од овие табели мора да биде во меморија.

Ако не се користат повеќенивоски табели, тогаш табелата на страници на овој систем ќе има големина од ставки.

1. Прво ниво се состои од: 1 табела, секогаш полнуваме со 1 табела
Прво ниво ставки: 32, имаш 5 бита на прво ниво, $2^5 = 32$

Второ ниво има 32 табели, имавме 32 стави на претходна табела.
Второ ниво ставки: 32, имаш 5 бита на второ ниво, $2^5 = 32$

Повеќенивоски табели ќе имат: 1024 ставки, $32 \times 32 = 1024$

7) Заокружување

Question 2
Not yet answered
Marked out of 1.00
Flag question

Кое од наведените полиња во табелата на страници се користи за да се определи дали одредена страница е користена во скоро време?

Select one:

- ☐ a. Protection
- ☐ b. Present/absent
- ☐ c. Caching disabled
- ☒ d. Referenced
- ☐ e. Modified

Question 4
Correct
Mark 1.00 out of 1.00
Flag question

Thrashing

Select one:

- ☐ a. е метод за отстранување на неискористени сектори на диск
- ☐ b. е основна карактеристика на системите со виртуелна меморија
- ☐ c. е подсистем за бришење на датотеки и нивно враќање од страна на корисникот
- ☒ d. може да се реши со суспендирање на некои од процесите кои се моментално активни

The correct answer is: може да се реши со суспендирање на некои од процесите кои се моментално активни

Question 4

Not answered

Marked out of 2.00

Flag question

Кај алгоритмот Clock Page Replacement, во моментот кога се случува грешка (page fault), стрелката покажува на страница со $R=1$. Што ќе се случи?

Select one:

- ☒ R ќе биде поставен на 0 и стрелката ќе биде поместена на следната позиција
- ☐ Страницата ќе биде исфрлена и стрелката ќе остане на истата позиција
- ☐ Страницата ќе биде исфрлена и стрелката ќе се помести на следната позиција
- ☐ R ќе биде поставен на 1 и стрелката ќе биде поместена на следната позиција

Question 5

Partially correct

Mark 0.40 out of 2.00

Flag question

Зголемување на големината на страницата ќе предизвика:

Select one or more:

- ☒ Намалување на табелата на страници
- ☒ Зголемување на TLB
- ☒ Намалување на TLB
- ☒ Зголемување на табелата на страници
- ☒ Зголемување на внатрешна фрагментација
- ☒ Намалување на внатрешна фрагментација

Кои од наведените искази се точни?

Select one or more:

- ☒ a. За да се најде ставка во TLB, мора да се споредуваат сите негови ставки поединечно ✓
- ☒ b. Показувач кон табелата на страници се чува во PCB ✓
- ☐ c. Табелата на страници се чува во PCB
- ☐ d. Користење на TLB го зголемува времето на мапирање виртуелна во физичка адреса
- ☐ e. TLB ги содржи сите ставки од табелата на страници

Thrashing настанува кога големината на физичката меморија е помала од бројот на страници кои ги користата процесите. Во овој случај, искористеноста на CPU се намалува. Наједноставно решение на проблемот е убивање на процес.

1. Датотечни Системи (21 - 40|v|)

1)

Нека еден датотечен систем има 64 000 блокови. Секој блок има големина од 200 байти и адреса од 2 байти. Доколку се користи евиденција за слободни блокови во листа, тогаш еден блок има капацитет да смести информација за **100** (цел број) слободни блокови. Според тоа, ако сите блокови се слободни, целата листа на слободни блокови би зафаќала **640** (цел број) блокови, а ако е слободен само 50 блокови, тогаш листата ќе зафаќа **1** (цел број) блок. Ако се користи евиденција за слободни блокови со бит-мапа, тогаш, во првиот случај, мапата ќе зафаќа **40** (цел број) блокови, а во вториот случај ќе зафаќа **40** (цел број) блокови.

1. Golemina na sekoj blok: 200 bajti
adresa na sekoj blok: bajti
 $200 \text{ bajti} / 2 \text{ bajti} = 100 \text{ slobodni blokovi}$
2. Vкупно 64 000 blokovi
 $64\,000 \text{ blokovi} / 100 \text{ slobodni blokovi} = 640$
3. Eden blok ima kapacitet da smesti info za 100 slobodni bloka
ako iame 50 slobodni bloka potrebno ni e samo 1 blok
4. Golemina na blok = 200B = 1600 bita -> pretvaras go od bajt vo bit pa go delish so vkupniot broj na blokovi
 $64\,000 / 1600 = 40 \text{ zafaka bit-mapata i vo dvata slucaja}$

2)

Дадена следнава MFT табела и првиот MFT запис за датотеката 'os.txt' е 100.

105	105	i	115	j	120	k	135	l	140	m
104										
103	50	d	70	e	90	f	95	g	100	h
102	250	34	340	100	660	1	700	5	800	9
101										
100	MFT103	MFT105	10	a	17	b	23	c		

Притоа, со зелена боја се дадени дополнителните MFT записи за датотеката, додека со жолта боја дадени се почетните блокови од датотеката, додека со буквите од a до m означени се бројот на блокови од наведениот почетен блок (односно run length), вредностите за run length се дадени во следнава табела:

a	b	c	d	e	f	g	h	i	j	k	l	m
4	2	7	4	1	4	3	3	2	3	10	1	12

Според дадената информација одговорете на следниве прашања:

- Стапува збор за **резидентен** запис.
- Коку записи од MFT табелата се резервирани за os.txt датотеката? **3** записи (внесете цел број).
- Доколку големината на блокот 2 KB, тогаш големината на датотеката **112** KB.

1. Bitno:
Rezidenten zapis: ako ja sobere datotekata samo vo eden MFT zapis.
Nerezidenten zapis: ako ja sobere vo dva ili povekje.
prvot zapis e vo 100.

Gledaj kaj redica 100 vo tabelata, ima uste 2 MFT zapisi oboeni zeleni.

Vкупно MFT zapisi: 3
Zapisot e nerezidenten.

gledas koi zapisi se za os.txt -> tie gi sodrzat bukvite od a do m.

100, 103 i 105. -> na mestoto od goleminite imame bukvi

sega gi sobiras goleminite sto ni se dadeni vo vtorata tabela ispod bukvite.

 $4 + 2 + 7 + 4 + 1 + 4 + 3 + 3 + 2 + 3 + 10 + 1 + 12 = 56$

Zbirot go mnozes: $56 * 2KB = 112KB$

3)

Датотеката `/home/teacher/exam.txt` има i-node со број 9701.

Процесот со PID=80 го извршува следниот код:

```
int fd1=open("/home/teacher/exam.txt",O_RDONLY); //open for reading
read(fd1,buffer1,150); //read 150 bytes in buffer1
int fd2=open("/home/teacher/exam.txt",O_RDONLY);
read(fd2, buffer2, 250); //read 250 bytes in buffer2
```

Процесот со PID=81 го извршува следниот код:

```
int fd3=open("/home/teacher/exam.txt",O_RDONLY); //open for writing
read(fd3, buffer3, 300); //read 300 bytes in buffer2
```

Процесот со PID=90 го извршува следниот код:

```
In /home/teacher/exam.txt /home/assistant/os.txt //create hard link (in source_file link)
In /home/teacher/exam.txt /home/dean/os.txt //create hard link
```

Пополнете ја содржината на дадените табели по извршување на кодот од процесите. Ако некои од полињата не се потребни, оставете ги празни.

Process table

PID=80

Index	Value
0	/
1	/
...	
7	
8	
9	

PID=81

Index	Value
0	/
1	/
...	
5	
6	
7	

PID=90

Index	Value
0	/
1	/
2	/
3	
4	
5	

System File Table

Address	Name	Value
2000	mode	+
	offset	
	ref.count	
	i-node ptr.	
	...	
2040	mode	+
	offset	
	ref.count	
	i-node ptr.	
	...	
2080	mode	+
	offset	
	ref.count	
	i-node ptr.	
	...	
20C0	mode	+
	offset	
	ref.count	
	i-node ptr.	

i-node Table

Address	Name	Value
B000	number	9701
	link count	
	ref. count	
	...	
B400	number	9639
	link count	
	ref. count	
	...	
B800	number	6760
	link count	
	ref. count	
	...	
BC00	number	8385
	link count	
	ref. count	
	...	

TABELITE:

okej, odeme po red so procesile

exam.txt ima i-node 9701.

Proces PID = 80 -> zabelezuje deka ovoj proces samo go otvara fajlot exam.txt za citanje (read), cita 150 bytes vo buffer1, pak go otvara fajlot po vtor pat samo za citanje i cita 250 bytes vo buffer2.

Sega gledame vo i-node tabelata, adresata za i-node na exam.txt e B000.

Sledno gledame na System File tabelata kaj adresa 2000 i 2040.

Za prvoto citanje:

mode: r (read) -> ova go odredivme na pocetokot
offset: 150 -> ova go odreduvame kaj linijata: read(fd1,buffer1,150)
ref count: 1 -> ovde sekogas e 1!
i-node ptr: B000 -> ova adresa ja odredivme vo i-node tabelata.

Za vtoroto citanje:

mode: r (read) -> ova go odredivme na pocetokot
offset: 250 -> ova go odreduvame kaj linijata: read(fd2,buffer2,250)
ref count: 1 -> ovde sekogas e 1!
i-node ptr: B000 -> ova adresa ja odredivme vo i-node tabelata.

Adresite 2000 i 2040 gi zapisuvame vo proces tabelata so PID=80 edno po drugo (BITNO!).

Proces PID=81 -> zabelezuje deka ovoj proces samo go otvara exam.txt za citanje i cita 300 bytes vo buffer3.

Sledno gledame na System File tabelata kaj adresa 2080.

Zapisuvame:

mode: r (read) -> ova go odredivme na pocetokot
offset: 300 -> ova go odreduvame kaj linijata: read(fd3,buffer3,300)
ref count: 1 -> ovde sekogas e 1!
i-node ptr: B000 -> ova adresa ja odredivme vo i-node tabelata.

Adresata 2080 ja zapisuvame vo proces tabelata so PID=81.

Proces PID=90, ovoj proces prave 2 hard links 🤔🤔

Bitno:

ln -> hard link

ln -s -> soft link

prvot argument e fajlot kon kogo se POKAZUVA

vtoriot e onoj na koj mu go menuvame POKAZUVACOT

Gledame prva linija:

assistant/os.txt POKAZUVA sega kon i nodeot na exam.txt -> bidejki e vtor argument go smenivme pokazuvacot.

bidejki e HARD link sega kaj i-node za exam.txt link count se inkrementira za 1 (samo vazi za HARD link)

Gledame vtorata linija:

Isto taka prave hard link kako prvata linija, inkrementiraj link count za 1.

Link count: 2 (sto inkrementiravme) + 1 (od samiot file) = 3 -> ova go zapisuvam vo i-node tabelata.

Ref count: 3 -> ova brojka dovagja od so go otvaravme fajlot 3 pati. ova go zapisuvam vo i-node tabelata.

Drugite polinja vo tabelata prazni gi ostavas ili " / " zavisi kako ti kazat da gi ostaves

2.

Preidnost na fd1, fd2 i fd3 da bide 7, 8 i 5, soodvetno.

Gledas vo proces tabelata:

fd1 e adresa 2000, indeksot na ovaa adresa e 7

fd2 e adresa 2040, indeksot na ovaa adresa e 8

fd3 e adresa 2080, indeksot na ovaa adresa e 5

3.

imenikot /home/teacher/ se go sodrzi parot ime na datoteka exam.txt - broj na i-node: 9701 - 4
imenikot /home/assistant/ se go sodrzi parot os.txt - 9701 - 4

imenikot home/teacher go sodrzi exam.txt vo nego.

i-node za nego ni e dadeno 9701.

imenikot home/assistant go sodrzi os.txt vo nego.

i-node za nego e isto 9701.

4.

Dokolku procesot so PID=70, namesta vtorata linija od prvot kod, go izmenuva kodot li -> /home/teacher/exam.txt /home/dean/os.txt (ovako soft link)
Togor link count za soodvetniot i-node se zmanuva vrednost 2 -> i imenikot /home/dean/ se go sodrzi parot os.txt - /home/teacher/exam.txt

ovde pisuva deka proces so pid=70, a takov ne postoe. Pa pretpostavuvam deka pid=90 bidejki tamo praveme links.

Ja gledame vtorata linija kaj pid=90: sega treba da prave soft link namesto hard.

soft link ne se broi vo link count.

soft link e samo apsolutna pateka kon originalniot file, a hard link e celosna kopija koja pokazuva kon istiot i-node

soft link count = 1 (abs path) + 1 (od samiot file) = 2

imenikot home/dean ke go sodrzi os.txt i apsolutnata pateka -> home/teacher/exam.txt

4)

Edna datotечен sistem koristi i-nodes strukturi kon imaat po 20 direktni pokazuvaci, 1 edinichen, 1 dvoen i 2 trojni indirektni pokazuvaci. Pretpostavete deka goleminata na blokot e 500 bajti, a adresata na sekoj blok na diskot e so golemina od 4 bajti. Sledno, toa,

- so lemovi na direktnite pokazuvaci moze da se pristapi do vkuпно $20 * 500 = 10000$ podatoci
- so eden edinichen indirektni blok moze da adresira vkuпно $500 / 4 = 125$ blovi,
- so eden dvoen indirektni blok moze da se adresira vkuпно $125 * 2 = 250$ blovi,
- so eden trojen indirektni blok moze da se adresira vkuпно $125 * 2 * 2 = 500$ blovi

Vkupniot broj na blokovi kon mozat da se adresirava so dadeniot i-node iznesuva $20 + 125 + 250 + (2 * 500) = 1953$ blovi

Dokolku pretpostavete deka vo keikat na imena (name cache) se naoga samo brojot na indirektni blovi na prvot imenikot i nema nista drugo i deka vo toot imenikot ima mnogu malu zaloz, togor potrebni se vkuпно 6 disk citanja za da se pročeta blokot 100 od datotekata /bar/foo/os.txt?

Kolku indirektni blokovi (blokovi so pokazuvaci) se potrebni za datoteka so golemina od 5KB (prizov 1KB=1000B), ne vkuпуvajući go samiot i-node. Odgovor: 6 indirektni blokovi.

1. Imas 20 direktni pokazuvaci i goleminata na blokot e 500 bajti.

Moze da se pristapi do: $20 * 500 = 10000$ podatoci

2. Golemina na blok: 500 bajti
Adresata na sekoj blok: 4 bajti

So eden EDINECEN indirektni blok moze da se adresira vkuпно: $500 \text{ bajti} / 4 \text{ bajti} = 125$

3. EDINECEN indirektni blok: 125 blokovi

So eden DVOEN indirektni blok moze da se adresira vkuпно $125 * 2 = 250$ blokovi

So eden TROEN indirektni blok moze da se adresira vkuпно $125 * 3 = 375$ blokovi

4. Vkupniot broj na blokovi e sumata na site ovie vrednosti plus direktnite pokazuvaci.

BITNO: imame 2 TROJNI indirektni blokovi, znaci se mnozat po 2 pred da se sumirat.

$20 + 125 + 250 + (2 * 1953) = 3906$

5. Treba da se pročeta blokot 100 od datotekata /bar/foo/os.txt

Prvo gledame dali spagja vo direktnite blokovi. Ne spagja bidejki tie se od 0-19, a treba 100 da citame.

Vtoro gledame dali spagjame vo indirektnite EDINICEN blokovi. DA, spagjame bidejki se adresirat 125 bloka a treba da citame 100, ako NE spagjavme preminuvame na DVOJNITE da provereme itn.

Treto sumirame:

1 citanje za root ili " / " +
1 citanje za bar +
1 citanje za foo +
1 citanje za os.txt +
1 citanje za ind EDINECEN +
1 citanje za data = 6 disk citanjta

6. Imame 5KB = 5000B datoteka, mozeme da ja smesteme vo 10 direktni blokovi, bidejki 1 e so golemina 500B.

Cim se smestuva vo direktni ni treba 0 indeksni blokovi.

Ako ne se smestuvase vo direktnite i otide vo indirektnite togas broeme kolku bloka ni trebata.

1. Дискови & SSD (41 - 50lv)

1)

Претпоставете дека секој од системите RAID 1 и RAID 5 кои ги поседува една компанија имаат по 8 диска и сите дискови имаат иста големина. Одговорете:

Колку место за складирање на податоци е достапно за секоја од конфигурациите (внесете цел број)?

• RAID 1 диска RAID 5 диска.

За секоја од конфигурациите, кој е минималниот број на оштетени дискови за да **МОЖЕ** да има губење на податоците?

• RAID 1 диска RAID 5 диска.

За секоја од конфигурациите, кој е минималниот број на оштетени дискови за **СИГУРНО** да има губење на податоци?

• RAID 1 диска RAID 5 диска.

Give your reasons

1. RAID(Shadow Legends)

Bitno:
n: 8 diska so ista големина

Prvo prasanje:

Raid1 = $n / 2 = 4$ -> место за складирање е половина од dostapните diskovi

Raid5 = $7 \rightarrow n - 1$ -> bidejki se trose $1/n$ za XOR blokovi za parnost sto go namaluva prostorot za 1 disk

Vtoro prasanje:

Raid1 = 2 -> se nastanuva zaguba ako eden disk i negoviot soodveten backup se ostetat.

Raid5 = 2 -> moze da gi presmeta i vrati zagubените podatoci od samo eden disk bidejki moze da presmeta samo edinicna parnost.

Treto prasanje:

Raid1 = 5 -> $(n/2) + 1$ diska za SIGURNO ke nastane zaguba.

Raid5 = 2 -> isto kako vtoroto prasanje.

2)

Дадена е состојбата на дел од SSD во кој се сместени податоци за датотеките 1, 2, 3 и 4. Секое непразно поле ја означува датотеката чишто податоци ги содржи. Која ќе биде содржината на секое поле од блокот C ако се избрише датотеката 4? Ако има потреба, празните полиња да се означат со вредност 0.

Block A			Block B		
1	1	1	2	2	2
1	1	1	2	2	2
1	2	2	2	3	3
2	2	2	3	3	3
Block C			Block D		
			3	3	3
			3	3	3
			3	3	4
			4	4	4

Block C

3	3	3
3	3	3
3	3	0
0	0	0

3) Заокружување

Кое од следниве правила важи кај SSD?

Select one:

- ☐ a. Читањето е 10 пати поскапо од бришењето, бришењето е 10 пати поскапо од запишувањето
- ☐ b. Читањето е 10 пати поскапо од запишувањето, бришењето е 10 пати поскапо од читањето
- ☐ c. Запишувањето е 10 пати поскапо од бришењето, бришењето е 10 пати поскапо од читањето
- ☒ d. Запишувањето е 10 пати поскапо од читањето, бришењето е 10 пати поскапо од запишувањето

[Clear my choice](#)

Give your reasons

Со помош на која информација по low-level форматирање на дискот се означува почеток на сектор

Select one:

- ☒ a. Sector Number

Question 4

Incorrect

Mark 0.00 out of 2.00

Flag question

Edit question

Кај кој тип на RAID е најлесно опоравувањето и враќањето на податоци по пад на диск?.

Select one or more:

- ☒ 1
- ☒ 2 ✗
- ☐ 3
- ☒ 4 ✗
- ☐ 5
- ☒ 10

Your answer is incorrect.

The correct answers are: 1, 10

Question 2

Not yet answered

Marked out of 1.00

Flag question

Кoj ro вчитува Master Boot Record-от кај еден компјутерски систем?

Select one:

- ☐ a. Соодветниот индексен јазел
- ☐ b. NTFS партицијата
- ☒ c. BIOS-от
- ☐ d. Јадрото на оперативниот систем