M E M O R A N D U M

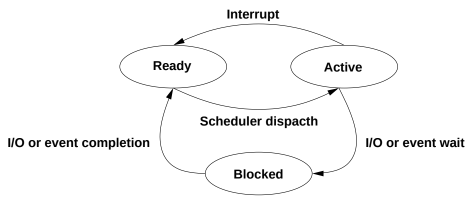
Universiteit van Stellenbosch
Departement Rekenaarwetenskap

Kursus: RW 314

Eksamen, Junie 2003 Tydsduur: 3 uur Eksaminator: J. J. Eloff

1. ’n Proses se toestand kan verander tussen “gereed”, “aktief”, en “geblokkeer”. Teken ’n diagram om aan te toon watter toestandsveranderinge moontlik is en watter gebeurtenis elke toestandsverandering sal veroorsaak.

   The state of a process can change between “ready”, “active”, and “blocked”. Draw a diagram to show which state changes are possible and what event will cause each state change.



6

2. Verduidelik kortliks wat semafore is. Lig u verduideliking toe met ’n voorbeeld wat die gebruik daarvan illustreer om die kritiese seksie probleem op te los.

   Briefly explain semaphores. Give an example that illustrates how semaphores can be used to solve the critical section problem.

A semaphore is an integer variable that can, apart from initialization, only be accessed through two operations **P** (test) and **V** (increment). Both P and V execute indivisibly to obtain mutual exclusion. Semaphores can either be implemented as spinlocks, or a queue mechanism can be used to increase CPU utilization. Note that simply using semaphores do not guarantee mutual exclusion since executing P and V incorrectly may still result in either deadlocks or race conditions.
Example: Consider two processes P_0 and P_1 and a single semaphore `s`. Both P_0 and P_1 might have the following structure:

```
do {
  P(s);
  /* critical section  */
  V(s);
  /* remainder section */
} while (1);
```

Initially `s` will be initialized to 1. Because P and V are indivisible, only one process will be allowed to enter its critical section. The other process will continue to evaluate the semaphore until the first process executes `V(S)`, thereby incrementing `s` and allowing another process to proceed.

8

3. Beskryf kortliks die verwantskap tussen logiese, lineêre en fisiese geheue-adresse.

   Briefly describe the relationship between logical, linear and physical memory addresses.

   Logical addresses are generated by the CPU while physical addresses are generated by the memory unit. A linear address is an intermediary address, often generated from the logical address when using segmentation and will eventually be translated into a physical address.

   3

4. Verduidelik kortliks die verskil tussen interne en eksterne fragmentasie ten opsigte van primêre geheue.

   Briefly explain the difference between internal and external fragmentation with regards to primary memory.

   Gaps are created in memory as processes are loaded and removed from it. Eventually the memory is broken into tiny fragments that in total can satisfy the memory requirements of a process, but is not contiguous. This is called external fragmentation. Internal fragmentation occurs when a block of memory is allocated, but the difference between the available block and the amount requested is too small to warrent keeping track of it. In stead, the operating system will allocate the whole block.

   4

5. Klokke is onontbeerlik vir die funksionering van baie bedryfstelsels. Noem **drie** belangrike gebruike van 'n klok in 'n tyddeel-stelsel.

   Clocks are essential to the operation of many operating systems. Name **three** *important uses of a clock in timesharing systems.*

   - Track the time of day
   - Time slicing in preemptive schedulers such as round-robin
   - Timing specific events when working with potentially unreliable devices

   3

6. Beskryf kortliks hoe 'n bedryfstelsel wat uitruiling ondersteun sal optree wanneer 'n proses 'n bladsyfout genereer. Lig u verduideliking toe met 'n voorbeeld.

   Briefly describe how an operating system that supports swapping will react when a process generates a page fault. Give an example to illustrate your answer.

   Assume that process P_1 is currently executing. Eventually a memory reference is made that results in a page fault exception. When the exception is generated, control is transferred to the operating system and the context of P_1 is saved. The operating system will examine the page tables to determine if the memory reference was valid. If not, the process will most likely be terminated. If the memory reference was valid, the page must be retrieved from secondary storage. If enough page frames are available, a frame is allocated, the page tables of P_1 are updated and the instruction that caused the fault is restarted. If there are not enough page frames available, the pager must select a candidate for replacement using a predetermined algorithm such as LRU. The victim page can either be local (one of P_1's own frames) or global (another process, say P_2).The data in the candidate page is written to secondary storage if its contents has changed (this can be determined by examining the 'dirty' bit of the victim page) before allocating the frame to P_1. The instruction that caused the fault is now restarted.

   8

7. 'n Proses genereer die volgende reeks bladsyverwysings tydens uitvoering: 2(R), 3(R), 2(W), 1(W), 5(R), 2(R), 6(R), 5(R), 3(W), 2(W), 5(R), 2(R) waar W verwys na 'n skryf operasie in die bladsy en R verwys na 'n lees operasie vanuit die betrokke bladsy. Die bedryfstelsel kan net vier bladsye op enige gegewe oomblik in geheue stoor vir elke proses. Beantwoord nou die volgende vrae met verwysing na die voorafgenoemde proses se gedrag.

A process generates the following sequence of page references while executing: 2(R), 3(R), 2(W), 1(W), 5(R), 2(R), 6(R), 5(R), 3(W), 2(W), 5(R), 2(R), where W indicates a write operation to a page and R indicates a read operation from a page. The operating system can only store four pages in memory at any given time for a process. Answer the following questions with regards to the preceeding behaviour of the process.

(a) Veronderstel die bedryfstelsel gebruik 'n strategie van aandringpaginering. Wat is die trefkoers indien 'n optimale bladsyvervangingsalgoritme gebruik word? Toon die inhoud van die bladsytabel na die laaste geheueverwysing.

Assume that the operating system employs a strategy of demand paging. What will the hit ratio be if an optimal page replacement algorithm is used? Show the contents of the page table after the last memory reference.

The hit ratio is $\frac{7}{12} = 0.58$.

2	3	2	1	5	2	6	5	3	2	5	2
2	2	2	2	2	2	2	2	2	2	2	2
	3	3	3	3	3	3	3	3	3	3	3
			1	1	1	6	6	6	6	6	6
				5	5	5	5	5	5	5	5

4

(b) Hoeveel bladsyfoute word gegenereer deur die proses indien die verbeterde tweedekans algoritme gebruik word in samewerking met die aandringpaginering strategie? Toon die inhoud van die bladsytabel, sowel as die bladsy klasifikasie wat met elke bladsy assosieer word na die laaste geheueverwysing.

How many page faults are generated by the process if the enhanced second-chance algorithm is used in conjunction with the demand paging strategy? Show the contents of the page table as well as the page classes associated with each entry after the last memory reference.

There are 8 page faults.

2 (R)			3 (R)			2 (W)			1 (W)			5 (R)			2 (R)		
R	M	P	R	M	P	R	M	P	R	M	P	R	M	P	R	M	P
1	0	2	1	0	2	1	1	2	1	1	2	1	1	2	1	1	2
0	0	0	1	0	3	1	0	3	1	0	3	1	0	3	1	0	3
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	5	1	0	5

6 (R)			5 (R)			3 (W)			2 (W)			5 (R)			2 (R)		
R	M	P	R	M	P	R	M	P	R	M	P	R	M	P	R	M	P
0	1	2	0	1	2	0	1	2	1	1	2	0	1	2	1	1	2
1	0	6	1	0	6	0	0	6	0	0	6	1	0	5	1	0	5
0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
0	0	5	1	0	5	1	1	3	1	1	3	1	1	3	1	1	3

R = Referenced   M = Modified   P = Page number

5

(c) Wat sal die trefkoers wees indien die stelsel 'n LRU algoritme kombineer met 'n strategie van vooruitpaginering deur die eerste drie bladsye (1, 2 en 3) in geheue te plaas voordat die proses begin uitvoer?

What will the hit ratio be if the system uses an LRU algorithm and employs a strategy of prepaging by loading the first three pages (1, 2 and 3) into memory before the process executes?

The hit ratio is $\frac{9}{12} = 0.75$.

2	3	2	1	5	2	6	5	3	2	5	2
1	1	1	1	1	1	1	1	3	3	3	3
2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	6	6	6	6	6	6
				5	5	5	5	5	5	5	5

2

8. Verduilelik kortliks die begrip van adresbinding en bespreek die tegniek van looptydbinding met spesifieke verwysing na die vereistes wat gestel word om dit suksesvol te ondersteun.

   Briefly describe the concept of address binding and discuss the use of execution (runtime) binding with specific reference to the requirements that must be met to successfully support it.

   > Address binding describes the process of assigning addresses to symbolic memory references and can be done at compile time, load time or during runtime. Runtime binding provides the most flexibility, since it allows the operating system to move processes from one position to another in memory. However, special hardware is required to support this because the logical and physical address space often differ. Most memory management units (MMUs) typically contain a *relocation* register. The contents of the relocation register is added to every logical address generated by a process to form a physical address. Paging can also be used to support runtime binding.

8

9. Definieër die term sparteling. Bespreek kortliks hoe die bladsyfout frewkensie model gebruik kan word om hierdie probleem aan te spreek.

   Define the term thrashing and briefly describe the application of the page fault frequency model as a possible solution to this problem.

   > Thrashing occurs when a process spends more time on paging than it does executing. The page fault frequency (PFF) model aims to prevent thrashing by controlling the page-fault rate. The model specifies an upper and lower bound and, depending on the page-fault rate, it adjusts the page frames allocated to a specific process in an attempt to bound it between these limits. In some cases, processes can be suspended to make additional frames available to processes with a high page fault rate.

6

10. Verduidelik kortliks die verskil tussen laevlak en logiese formatering.

    Briefly describe the difference between lowlevel and logical formatting.

    > Lowlevel formatting initializes the physical structure of a disk by dividing it into sectors and initializes every sector with certain information such as the sector number and ECC values. Logical formatting creates the file system by initializing specific sectors with certain structures such as a FAT or i-node table.

4

11. 'n Werkstasie bevat kasgeheue, primêre geheue en virtuele geheue wat op 'n skyf gestoor word. Dit neem 20ns om 'n adresverwysing vanaf die kasgeheue op te los. Indien dit nie in die kasgeheue voorkom nie, word 60ns benodig om die woord vanaf primêre geheue na die kasgeheue oor te dra waarna die adresverwysing weer uitgereik word. Indien die woord nie in die primêre geheue voorkom nie, word 12ms benodig om dit vanaf sekondêre geheue na primêre geheue oor te dra. 'n Addisionele 60ns is nodig om die woord van primêre geheue na die kasgeheue oor te dra voordat die adresverwysing weer uitgereik word. Gestel dat die trefkoers vir die kasgeheue 0.8 is en die trefkoers vir primêre geheue 0.7 is. Wat is die gemiddelde toegangs tyd om 'n woord te lees of skryf? Druk u antwoord uit in terme van nanosekondes.

    A workstation contains cache memory, primary memory and virtual memory residing on a disk. It takes 20ns to access a word in the cache. If the word can not be found in the cache, 60ns are required to retrieve the word from primary memory and store it in the cache, afterwhich the reference is started again. If the word does not reside in primary memory, 12ms are required to retrieve the word from virtual memory followed by an additional 60ns to transfer the word into cache memory. The cache hit ratio is 0.8 and the primary memory hit ratio is 0.7. What is the average time, in nanoseconds, required to access a word?

- Access time from cache memory: 20ns
- Access time from primary memory: $20 + 60 = 80$ns
- Access time from secondary memory: $12 \times 10^6 + 60 + 20$ns
- Average access time: $(0.8 \times 20) + 0.2((0.7 \times 80) + (0.3 \times ((12 \times 10^6) + 80)))$
- Average access time: 720032ns

6

12. Verskaf 'n kort beskrywing van die werking van 'n stelseldrywer vir skyfeenhede.

    Give a brief description of the functioning of a disk driver.

    A disk driver would typically make its interface available to user-level processes through blocking system calls. When a new request arrives, the calling process is blocked and the request is either sent directly to the disk or placed into a waiting queue. If the request can be processed, the necessary parameters are sent to the disk and control is transferred to the scheduler to select a new process. Once the disk operation is completed, an interrupt is generated by the controller to inform the driver that the previous request has been serviced. Upon receiving the interrupt, control is transferred back to the device driver which will examine the results of the completed operation (possibly reporting errors back to the user) and copy the data from the disk buffer to the buffer originally specified by the process. The process will be unblocked and the next request in the driver's waiting queue will be processed before transferring control to the scheduler.

7

13. Verduidelik kortliks die verskil tussen sektorsparing en sektorverglying ten op sigte van die bestuur van beskadigde skyfblokke.

    Briefly describe the difference between sector sparing and sector slipping with regards to managing damaged disk blocks.

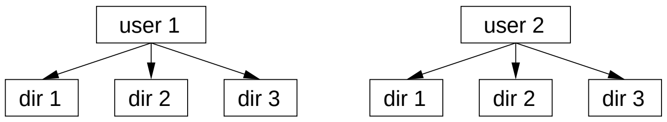
    Sector sparing refers to the replacement of disk blocks by remapping the damaged disk block on to a spare disk block. Sector slipping entails adjusting the logical block numbers by sliding them upwards. For example, if block 10 is damaged, and the first available block is 20, then block 20 would be allocated on a spare block, while the other blocks are adjusted so that block 19 is moved to block 20, 18 to 19, etc until block 10 is copied into block 11.

4

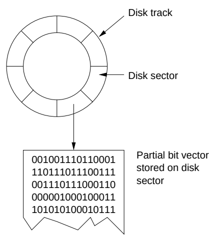
14. Die gebruik van lêerstelsels is 'n integrale komponent van meeste werkstasie-bedryfstelsels, veral in 'n multi-gebruikers omgewing. Verskaf 'n kritiese bespreking van hoe 'n lêerstelsel implementeer kan word met spesifieke verwysings na die bestuur van skyfblokke, die implementering van gidse, allokasie strategieë asook moontlike opsies wat gevolg kan word om effektiwiteit te verhoog. Lig u bespreking toe met diagramme en voorbeelde.

    File systems are an important component of most workstation based operating systems, especially within a multi-user environment. Give a critical discussion on how a file system can be implemented, with specific references to the free space management, implementing directories, allocation strategies and possible options to increase performance. Use diagrams and examples to illustrate your answer.

Directories: A number of options exist. Single level directories are fairly simple to implement and quite efficient, but creates problems when a filename is shared between users. This problem can be solved by implementing a two-level structure using the user's name as the first directory level. Better organization can be obtained by using a tree structure and a number of implementation options exist, such as acyclic graphs which provides the functionality to support file sharing. The figure below shows a two-level directory structure.



Free space: The simplest way to manage free space makes use of a bit vector. Depending on the size of the secondary storage device, a number of disk blocks may be reserved to store the bit vector. Although efficient, this method may require large amounts of memory when working with high capacity storage devices. Alternatively, a linked list can be kept to track free blocks and leads to less efficient search times. Other methods to track free space include grouping and counting. The figure below illustrates the bit vector method. Bits set to 1 indicate allocated blocks, while a 0 indicates an open (free) block.



Allocation: Depending on the file system's structure and use, disk space can either be allocated contiguously or allocated blocks can be linked. Apart from these options, other strategies such as first fit and best fit can be used to limit the amount of fragmentation. Unlike primary memory, compacting secondary storages is an extremely expensive task and should be avoided if possible.
Efficiency: A variety of methods can be implemented to improve the efficiency of file systems:

- Structures containing the root directory can be kept in memory to reduce physical disk accesses.
- A buffer cache can be implemented to hold additional disk blocks that are frequently accessed, furhter reducing disk accesses.
- Efficient data structures such as hash tables can be used to improve search times when working with large structures such as directories.
- Depending on the size of secondary memory, multiple sectors can be grouped into blocks. The system can then perform all its tasks in terms of blocks, reducing the size of almost all the data structures that must be maintained. However, this will increase the amount of internal fragmentation when working with small files.
- Seek times can be improved by using a disk scheduling algorithm such as SSTF or SCAN.

22

Totaal: / *Total:* 100