Department of Computer Science and Engineering
Chalmers

Exam in Unix Internals EDA203/DIT681

DAY: 2010-05-25 TIME: 14.00-18.00 PLACE: M building
==

Examiner: Arne Dahlberg

Questions during exam: Arne Dahlberg, 772 1705

Solutions: No solutions will be posted

Grading Policy:	EDA203	3: 30-38, 4: 39-47, 5: 48-60
	DIT681	G: 30-47, VG: 48-60

Aids during the exam:

- McKusick, Neville-Neil: The design and implementation of the FreeBSD operating system.
- An English language dictionary.

Instructions:

- Start answering each assignment on a new page; number the pages and use only one side of each sheet of paper.
- Justify all answers. Lack of justification can lead to loss of credit even if the answer might be correct.
- No references to the text book is allowed and copying of text from the book is not allowed.
- If you make any assumptions in answering any item, do not forget to clearly state what you assume.
- Write clearly! If I cannot read your solution, I will assume that it is wrong.

Problem 1 (10p)

a. The first step in *exec* is to reserve memory resources and check that enough virtual memory is available to execute the process. Explain how another inoccent process could be affected if this check was not done. (3p)

b. The *exec* system call will need to allocate new virtual memory segments for the process. Describe how this allocation is implemented. (4p)

c. If *exec* is called to start execution of a program (file) that is already executed by another process, the code segment shall be shared between the processes. Describe how the system can detect that the program is already executed by another process. (3p)

Problem 2 (10p)

a. Describe how a process group works. What is the most important reason why process groups were added to the system? Why is process groups collected into a session? Explain how a a process group can become an "orphaned process group". What will happen to the processes that are members in such an "orphaned process group" ? (6p)

b. A login-shell starts a process in the background and then exits (logging out). Explain the mechanism that prevents the remaining background process to read from the terminal. (4p)

Problem 3 (10p)

a. In most cases the buffer cache in FreeBSD use page frames to store the data blocks, but there are some exceptions. Explain why page frames are not always used. (2p)

b. When the *bread()* routine is used to look up a buffer in the buffer cache, the returned buffer is also mapped into kernel virtual memory. What is the reason that the returned buffer is mapped into kernel virtual memory? (2p)

c. In some cases there may exist buffers with valid content on the CLEAN list that have never been used. Give an example of how this could happen. (2p)

d. Explain the mechanism that is used to prevent a buffer that is repeatedly written by a program from beeing prematurely written to disk. (2p)

e. How can the kernel determine if a file block is already present in the buffer cache? (2p)

Problem 4 (10p)

a. NFS may have problems with nonidempotent operations. Give an example of such a problem. How is the problem with nonidempotent operations solved in FreeBSD? (2p)

b. NFS implemented according to version 2 protocol may have very bad write performance. How is this problem solved in the FreeBSD implementation of NFS? (2p)

c. In NFS version 3, UDP may be replaced by TCP as the transport protocol. Give at least two reasons for this change. (4p)

d. When running an NFS server, an extra server process called *portmap* is needed. What is the reason that the *portmap* server is needed. (2p)

Problem 5 (10p)

a. What is meant with an association when talking about network communication? (1p)

b. Describe how the implementation is done to allow for the correct association to be found when a packet is received. Also explain which data structures that are used. (4p)

c. In the network implementation mbufs are used to store data and addresses. They are specially designed to be flexible when adding data before or after other objects, but at the same time being efficient in the handling of large data blocks. Descibe how. (5p)

Problem 6 (10p)

a. Assume that a UDP packet arrives on an Ethernet. Describe the handling of this packet from it is received by the Ethernet interface till it is queued at the UDP socket. (8p)

b. How do the driver know that the packet in part a. shall be delivered to the *ip_input()* routine (and no other input routine)? (2p)