Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr. Frank Bellosa
Lukas Werling, M.Sc.
Dr. Marc Rittinghaus

Musterlösung Nachklausur
09.09.2021

Alle Punkteangaben ohne Gewähr!

- Bitte tragen Sie zuerst auf dem Deckblatt Ihren Namen, Ihren Vornamen und Ihre Matrikelnummer ein. Tragen Sie dann auf den anderen Blättern (auch auf Konzeptblättern) Ihre Matrikelnummer ein.

  Please fill in your last name, your first name, and your matriculation number on this page and fill in your matriculation number on all other (including draft) pages.

- Die Prüfung besteht aus 24 Blättern: Einem Deckblatt, 23 Aufgabenblättern mit insgesamt 5 Theorieaufgaben und 3 Programmieraufgaben sowie 0 Seiten Man-Pages.

  The examination consists of 24 pages: One cover sheet, 23 sheets containing 5 theory assignments as well as 3 programming assignments, and 0 sheets with man pages.

- Es sind keinerlei Hilfsmittel erlaubt!

  No additional material is allowed!

- Die Prüfung ist nicht bestanden, wenn Sie versuchen, aktiv oder passiv zu betrügen.

  You fail the examination if you try to cheat actively or passively.

- Sie können auch die Rückseite der Aufgabenblätter für Ihre Antworten verwenden. Wenn Sie zusätzliches Konzeptpapier benötigen, verständigen Sie bitte die Klausuraufsicht.

  You can use the back side of the assignment sheets for your answers. If you need additional draft paper, please notify one of the supervisors.

- Bitte machen Sie eindeutig klar, was Ihre endgültige Lösung zu den jeweiligen Teilaufgaben ist. Teilaufgaben mit mehreren widersprüchlichen Lösungen werden mit 0 Punkten bewertet.

  Make sure to clearly mark your final solution to each question. Questions with multiple, contradicting answers are void (0 points).

- Programmieraufgaben sind gemäß der Vorlesung in C zu lösen.

  Programming assignments have to be solved in C.

Die folgende Tabelle wird von uns ausgefüllt!
The following table is completed by us!

Aufgabe	T1	T2	T3	T4	T5	P1	P2	P3	Total
Max. Punkte	9	9	9	9	9	15	15	15	90
Erreichte Punkte									

Aufgabe T1: Grundlagen

Assignment T1: Basics

a) In einem herkömmlichen Betriebssystem findet eine Unterteilung der Ausführung in Benutzer- und Kernelmodus statt. Unter welchen Umständen kann es Sinn machen, auf eine solche Unterteilung zu verzichten? Geben Sie ein Beispiel an. **1 pt**

In a conventional operating system, the execution is separated into user and kernel mode. Under which circumstances can it make sense to abandon this separation? Give an example.

Solution:

Removing the separation into user and kernel mode can make sense in specialized environments such as on embedded devices or where only a single fixed application is run on the operating system (e.g., a library OS) **(1.0 P)**.

Nennen und begründen Sie einen Vor- und einen Nachteil einer gemeinsamen Ausführung von Anwendung und Betriebssystem im Kernelmodus. **2 pt**

Name and explain one advantage and one disadvantage of running the application and operating system together in kernel mode.

Solution:

An advantage of running the application together with the operating system in kernel space is a potentially faster execution **(0.5 P)** *as expensive system calls can be replaced with cheap function calls* **(0.5 P)**.

A disadvantage is that bugs in the application can easily corrupt operating system data structures **(0.5 P)** *and thereby reduce reliability and stability* **(0.5 P)**.

b) Welche Aufgabe kommt dem *System Call Dispatcher* zu? **1 pt**

What is the role of the system call dispatcher*?*

Solution:

The system call dispatcher performs the lookup in the system call table **(0.5 P)** *and jumps to the handler* **(0.5 P)**.

c) Multiprozessorsysteme können Interrupts üblicherweise an einen beliebigen Prozessor zustellen. Geben Sie zwei Gründe an, weshalb die Wahl des Zielprozessors die Performance des Systems beeinflussen kann. **2 pt**

Multi-processor systems are usually capable of delivering interrupts to an arbitrary processor. Give two reasons why the choice of the target processor may affect system performance.

Solution:

Valid reasons include:

- *If the interrupt is delivered to a busy processor instead of an idle one, the running process is unnecessarily interrupted and delayed* **(1.0 P)**.
- *Each processor has its own cache. How efficiently the interrupt can be handled depends on the data in the cache* **(1.0 P)**. *Processors which have recently accessed the data required to process the interrupt will perform significantly better than processors with a "cold" cache.*

Answers where both reasons target the processor utilization (e.g., blocking of other processes etc.) or efficient use of caches will be treated as redundant.

d) Das `.bss`-Segment einer ELF-Datei wird vergrößert. Welche Auswirkung hat dies auf die Dateigröße sowie auf den Verbrauch von virtuellem und physischem Speicher? Gehen Sie von Demand Paging aus. Begründen Sie Ihre Antwort kurz. **3 pt**

The `.bss` segment of an ELF file is enlarged. How does this affect the file size and the consumption of virtual and physical memory? Assume demand paging. Briefly justify your answer.

Solution:

File size *Does not change* **(0.5 P)** *because, being all zeros, the `.bss` segment is not actually stored on disk* **(0.5 P)**.

Virtual memory *Increases according to the change of the segment size* **(0.5 P)** *because the segment has to be represented in virtual memory to allow accesses* **(0.5 P)**.

Physical memory *May increase* **(0.5 P)**, *depending on if the newly reserved space is actually accessed. Also depends on other paging decisions (e.g., page replacement)* **(0.5 P)**.

Total: 9.0pt

Aufgabe T2: Prozesse und Threads

Assignment T2: Processes and Threads

a) Eine Kernfunktion des Betriebssystems ist die Prozessisolation. Was bedeutet das und wie wird diese Isolation realisiert? **1.5 pt**

A core functionality of the operating system is process isolation. What does that mean and how does the OS implement isolation?

Solution:

The operating system puts each process in its own address space **(0.5 P)** *and multiplexes hardware between all processes (limited direct execution)* **(0.5 P)**. *Consequently, processes cannot see each other and thus cannot read or modify other process's data* **(0.5 P)**.

Nennen Sie zwei Betriebssystemfunktionalitäten, durch die die Isolation zwischen Prozessen aufgeweicht wird. **1 pt**

Name two operating system functionalities that weaken the isolation between processes.

Solution:

- *IPC mechanisms such as shared memory, message passing, signals*
- *(shared) file system access*
- *shared resources after* `fork()` *such as file descriptors*
- *debuggers*
- *the* `/proc` *file system*

b) Was ist ein Daemon? Nennen Sie ein Beispiel. **1 pt**

What is a daemon? Give an example.

Solution:

A daemon is a process designed to run in the background, without direct control from the user. **(0.5 P)**

Examples:

- *networking software such as web servers*
- *hardware control software such as sound servers or network configuration agents*
- *long-term schedulers in user space such as the init process or cron*

Was ist der *init-Prozess*? Welche Rolle spielt er im Bezug auf Daemons? **1 pt**

What is the init process? *Which role does it play with regard to daemons?*

Solution:

The init process is the initial user space process started by the kernel. **(0.5 P)**

The init process is responsible for launching most daemons. **(0.5 P)** ***OR***

In the process hierarchy, daemons are usually placed directly below the init process. This can be accomplished by having the daemon's parent process exit. **(0.5 P)**

c) Zum Starten eines neuen Threads muss Speicher alloziert werden. Welche Strukturen benötigen diesen Speicher? In welchem Adressraum befinden sich die Strukturen bei Einsatz des One-to-One-Modells? **2 pt**

When starting a new thread, memory needs to be allocated. Which structures need this memory? In the one-to-one model, in which address space are these structures located?

Solution:

A thread needs memory for:

- *the stack* **(0.5 P)**, *allocated in the user address space* **(0.5 P)**
- *the thread control block (TCB)* **(0.5 P)**, *allocated in kernel space.* **(0.5 P)**

d) Welches Problem wird durch Virtual Round Robin im Vergleich zu normalem Round Robin gelöst? Erklären Sie Problem und Lösung. **2 pt**

Which problem does Virtual Round Robin solve in comparison with normal round robin? Explain the problem and the solution.

Solution:

RR is unfair to I/O bound jobs **(0.5 P)** *because they block before using up their time slice* **(0.5 P)**. *When a process blocks, Virtual RR stores the remaining time slice* **(0.5 P)** *and gives those processes priority over CPU-bound processes until the stored time slice is used up.* **(0.5 P)**

e) Nennen Sie einen Scheduling-Algorithmus, der nur auf reinen Batch-Systemen sinnvoll eingesetzt werden kann. **0.5 pt**

Name a scheduling algorithm which only makes sense on pure batch systems.

Solution:

- *First Come First Serve*
- *Shortest Job First*

Total: 9.0pt

Aufgabe T3: Speicher

Assignment T3: Memory

a) Bei einem *Pufferüberlauf* schreibt ein Programm in Richtung ansteigender Adressen über die Grenzen eines Puffers. Liegt der Puffer auf dem Stack, kann die Rücksprungadresse überschrieben werden. Lässt sich dies verhindern, indem die Wachstumsrichtung des Stacks verändert wird? Begründen Sie Ihre Antwort. **1 pt**

During a buffer overflow *a program writes over the boundaries of a buffer towards increasing addresses. If the buffer is allocated on the stack, the return address can be overwritten. Can this be prevented by changing the direction of stack growth? Justify your answer.*

Solution:

Generally, no. Although the program cannot overwrite the return address of the same or previous frames (relative to the frame that contains the buffer), it can still overwrite the return address of following frames **(1.0 P)**.

```
void myfunc(void) {
    char buffer[256];

    /* Buffer overflow in vulnerable_function() overwrites the return
     * address of vulnerable_function() */
    vulnerable_function(&buffer);
}
```

b) Diskutieren Sie Vor- und Nachteile von Bitmaps gegenüber einfach verketteter Listen (eine Seite pro Bit und Listenelement) zur Verwaltung freier Speicherseiten bzgl. Speicherverbrauch und algorithmischer Komplexität bei Allokation einer Seite. **2 pt**

Discuss the advantages and disadvantages of bitmaps versus singly-linked lists (one page per bit and list node) for managing free memory pages in terms of memory consumption and algorithmic complexity for allocating a page.

Solution:

Memory consumption *The bitmap is very compact (i.e., 1 bit per page), but the list nodes for free memory can usually be placed in the free memory itself* **(0.5 P)**. *So in this case the list can be more efficient* **(0.5 P)**.

Algorithmic complexity *Finding free memory in the bitmap is an* $\mathcal{O}(n)$ *operation with* n *being the number of pages in the system* **(0.5 P)**. *For a singly-linked list it is* $\mathcal{O}(1)$ **(0.5 P)**.

Other answers and conclusions accepted if plausible.

c) Speicher, der als interne Fragmentierung bezeichnet wird, darf nicht in der Frei-Liste auftauchen, wohingegen als externe Fragmentierung bezeichneter Speicher in der Frei-Liste auftauchen muss. **0.5 pt**

Memory designated as internal fragmentation must not appear in the free list, whereas memory designated as external fragmentation must appear in the free list.

Solution:

■ *Ja / Yes* □ *Nein / No*

d) Erläutern Sie den Begriff *Demand Paging*. Welchen Nachteil hat dieses Verfahren? **1 pt**

Explain the term demand paging. *What is the disadvantage of this method?*

Solution:

With demand paging pages in the virtual address space are actually allocated in physical memory and filled with contents only on their first access **(0.5 P)**. *A disadvantage is that programs may experience many page faults at startup, which might be less efficient from a performance perspective than doing pre-paging* **(0.5 P)**.

e) Gegeben sei ein System mit einer 3-stufigen Seitentabelle mit 128-bit-Einträgen und 512 KiB-Seiten. Unterteilen Sie die folgende virtuelle Adresse, wie es für die Adressübersetzung nötig ist. Geben Sie dabei für jeden Abschnitt die Anzahl der Bits an. **2 pt**

Assume a system with a 3-level page table with 128-bit entries and 512 KiB pages. Divide the following virtual address as needed for the address translation. For each part, provide the number of bits.

Solution:

We have $2^{19}/2^4 = 2^{15}$ *page table entries per 512 KiB* $= 2^{19}$ *bytes page. Since we have a 3-level page table, the virtual address is divided into three 15-bit offsets into the page tables and one 19-bit offset into the actual page.*

Virtual Address	*15 bits*	*15 bits*	*15 bits*	*19 bits*

Wie groß ist der virtuelle Adressraum in Bytes? Zweierpotenz ausreichend. **0.5 pt**

What size is the virtual address space in bytes? Power of two sufficient.

Solution:

$15 \times 3 + 19 = 64$ *bits for the virtual address* $\rightarrow 2^{64}$ *bytes* **(0.5 P)**

Wie könnte die Speicherorganisation und Adressübersetzung bei gleichbleibender Adressraumgröße prinzipiell angepasst werden, um für dünn besetzte Adressräume Speicher zu sparen (keine Rechnung nötig)? **1 pt**

How could memory organization and address translation be adapted in principle to save memory for sparse address spaces (no calculation necessary)? The size of the address space should remain the same.

Solution:

The page size could be reduced **(0.5 P)** *while increasing the number of levels* **(0.5 P)**. *Hint: For an inverted page table, it depends on the size of the physical memory. Also it is less likely to safe memory for very sparse virtual address spaces.*

Wie wirkt sich dies auf die Effektivität des TLB aus? Begründen Sie Ihre Antwort. **1 pt**

What impact does this have on the effectiveness of the TLB? Justify your answer.

Solution:

With smaller pages, the TLB reach is reduced **(0.5 P)**. *In consequence, the TLB hit rate may decrease* **(0.5 P)**, *depending on the working set of the program.*

Total: 9.0pt

Aufgabe T4: Koordination und Kommunikation von Prozessen

Assignment T4: Process Coordination and Communication

a) Welche der in der Vorlesung vorgestellten Anforderungen an Synchronisierungsprimitive wird durch eine starke Semaphore erfüllt, durch eine schwache aber nicht? **0.5 pt**

Which of the requirements for synchronization primitives presented in the lecture is fulfilled by a strong semaphore, but not by a weak semaphore?

Solution:

Bounded waiting **(0.5 P)**

Welchen Thread weckt eine starke Semaphore auf, um die Anforderung zu erfüllen? **0.5 pt**

Which thread does a strong semaphore wake up to fulfill this requirement?

Solution:

The thread which has been waiting for the longest time **(0.5 P)**.

b) Sie möchten das vom Kernel angebotene Message Passing zwischen den User-Level-Threads eines Prozesses verwenden. Weshalb muss hierbei mindestens entweder Senden oder Empfangen asynchron sein? **1.5 pt**

You want to use the message passing provided by the kernel between the user-level threads of a process. In this case, why does at least either sending or receiving have to be asynchronous?

Solution:

If both sending and receiving are synchronous and blocking, both the receiver and the sender thread have to execute a system call at the same time **(1.0 P)**. *However, user-level threading maps both threads onto one kernel thread, so only one can execute a blocking system call at a time* **(0.5 P)**.

Weshalb eignet sich kernelseitiges direktes Message Passing nicht dazu, einzelnen User-Level-Threads gezielt Nachrichten zu schicken? **1 pt**

Why is it not possible to use kernel-based direct message passing to send messages specifically to individual user-level threads?

Solution:

Direct message passing can only target individual processes (or potentially kernel-level threads). All user-level threads are, however, executed in the same process and kernel-level thread **(1.0 P)**.

c) Nennen Sie die vier Bedingungen für einen Deadlock. **2 pt**

Name the four conditions for a deadlock.

Solution:

Mutual exclusion **(0.5 P)**, *hold and wait* **(0.5 P)**, *no preemption* **(0.5 P)**, *and circular wait* **(0.5 P)**.

Gegeben sei ein System, in dem Threads nie auf ein Spinlock zugreifen, falls sie bereits ein weiteres mit einer höheren virtuellen Adresse halten. Welche Bedingung für Deadlocks zwischen den Threads eines Prozesses wird dadurch verhindert? Erklären Sie kurz, weshalb die Bedingung nicht auftreten kann. Gehen Sie davon aus, dass nur Spinlocks zur Synchronisierung verwendet werden und dass physischer Speicher nicht mehrfach in den Adressraum abgebildet wird. **1.5 pt**

Assume a system where threads never access a spinlock if they already hold another spinlock with a higher virtual address. Which condition for deadlocks between the threads of a process is prevented by this scheme? Briefly explain why the condition is impossible. Assume that only spinlocks are used for synchronization and that physical memory is not mapped multiple times into the same address space.

Solution:

The approach prevents circular wait **(0.5 P)**. *Any cycle requires one thread to hold a spinlock with a higher virtual address and wait for a spinlock with a lower address, whereas another thread has to hold a spinlock with a lower virtual add and has to wait for a higher address* **(1.0 P)**. *Such scenarios are not allowed by the given system.*

Mehrere Prozesse werden durch Spinlocks in geteilten Speicherbereichen synchronisiert. Warum kann der Ansatz in manchen Situationen Deadlocks zwischen den Prozessen nicht verhindern? **2 pt**

Multiple processes are synchronized by spinlocks in shared memory regions. Why is the approach in some situations unable to prevent deadlocks between the processes?

Solution:

The approach requires spinlocks to have the same virtual address order in all participating processes **(1.0 P)**. *Shared memory segments, however, do not have to be mapped in the same order in all participating processes* **(0.5 P)**, *so the order in which spinlocks can be aquired varies from process to process, allowing cycles to develop* **(0.5 P)**.

Total: 9.0pt

Aufgabe T5: I/O, Hintergrundspeicher und Dateisysteme

Assignment T5: I/O, Secondary Storage, and File Systems

a) Auf Unix-System werden häufig wie unten dargestellt getrennte Dateisysteme für Wurzel- und Homeverzeichnis eingesetzt. Ein Benutzer versucht, mithilfe des `ln`-Kommandos Dateiverknüpfungen anzulegen.

Unix systems are often configured to use separate file systems for the root and home directories, as shown below. A user tries to create links with the `ln` command.

```
$ mount
[...]
/dev/sda1 on / type ext4 (rw)
/dev/sda2 on /home type ext4 (rw)

$ ln /etc/passwd /home/user/link1    # (1)
$ ln -s /etc/passwd /home/user/link2 # (2)
```

Welche Art von Verknüpfung versuchen die beiden Kommandos anzulegen? **1 pt**

What kind of link do the two commands try to create?

Solution:

(1) hard link
(2) symbolic link

Sind die Befehle erfolgreich? Begründen Sie jeweils. Nehmen Sie an, dass die referenzierten Dateien und Ordner wie erwartet existieren. **2 pt**

Are the commands successful? Explain for both commands. Assume that the referenced files and directories exist as expected.

Solution:

Hard links are resolved by the file system (in this case, ext4) and thus have to be within one file system. **(0.5 P)** *Command (1) tries to create a hard link across different file systems, so it will fail.* **(0.5 P)**

Symlinks are resolved by the virtual file system (VFS) and reference paths. **(0.5 P)** *Consequently, command (2) will succeed.* **(0.5 P)**

b) Trotz des Aufkommens von Solid State Disks (SSD) sind Festplatten heutzutage weiter relevant. Nennen Sie drei Vorteile von Festplatten. **1.5 pt**

Even with the rise of Solid State Disks (SSD), hard disks are still relevant today. Name three advantages of hard disks.

Solution:

Hard disks are... **(0.5 P)** *per item)*

- *cost efficient / cheaper than SSDs*
- *endurable: number of reads and writes is nearly infinite*
- *reliable: SSDs have more uncorrectable data errors*
- *large: up to 20 TB per device, more than SSDs can provide*
- *simpler: no need for complex controller logic for FTL*

c) Nehmen Sie an, Sie betreuen eine Anwendung, die unter schlechter Performanz des Hintergrundspeichers leidet. Als Abhilfe erwägen Sie den Einsatz mehrerer Festplatten entweder als RAID 0-System oder als JBOD (Just a Bunch Of Disks), mit unabhängigen Dateisystemen pro Festplatte. Welche Variante würden Sie bevorzugen? Begründen Sie. **2 pt**

Suppose you are maintaining an application that is suffering from poor storage performance. As a solution, you consider using multiple hard disks either as a RAID 0 or as a JBOD (Just a Bunch Of Disks) with independent file systems per hard disk. Which variant would you prefer? Give reasons.

Solution:

Both answers are valid.

Reasons to prefer RAID 0:

- *There is no need to modify the application* **(0.5 P)** *as RAID systems are transparent to the software.* **(0.5 P)**
- *Higher storage performance than with JBOD especially for sequential accesses as consecutive blocks map to different devices.* **(1 P)**
- *All drives form a single volume, asymmetric disk usage is not possible.* **(0.5 P)**
- *Lower likelyhood of overloading a single drive than with JBOD.* **(0.5 P)**

Reasons to prefer JBOD:

- *No need for special RAID controllers* **(0.5 P)** *and thus cheaper to deploy.* **(0.5 P)**
- *Higher storage performance than with JBOD if the application can map parallel operations to different drives.* **(1 P)**
- *A drive failure does not affect data on the other drives.* **(0.5 P)**

d) Ein Dateisystem nutzt Indexed Allocation und kann in einem Indexblock wahlweise 64 Pointer auf fixe Fragmente speichern oder 32 Extents. Welche zusätzliche Information muss in einem Extent vermerkt sein? **0.5 pt**

A file system uses indexed allocation and can store either 64 pointers to fixed fragments in an index block or 32 extents. Which additional piece of information is stored in an extent?

Solution:

In addition to the block pointer, an extent also needs to store the length of the fragment.

Nennen Sie zwei Vorteile davon, Extents einzusetzen. **1 pt**

Name two advantages of using extents.

Solution:

Using extents...

- *improves contiguity*
- *reduces index size*
- *reduces overhead from unneeded pointers*

Note: *The maximum file size does not automatically improve with extents. File system developers generally choose a maximum file size and design their metadata (e.g., file size field, structure of indirect blocks) accordingly.*

e) Welche Aufgabe hat ein *fsck*-Programm? **1 pt**

What is the purpose of an fsck *program?*

Solution:

A file system checker verifies invariants of the file system metadata. It can fix certain types of metadata corruption and can thus prevent propagation of these issues while the file system is in use.

Total: 9.0pt

Aufgabe P1: C Grundlagen

Assignment P1: C Basics

a) In dem untenstehenden Code haben sich 7 Fehler eingeschlichen. Markieren Sie die fehlerhaften Zeilen mit einem X und korrigieren Sie den Code. Gehen Sie von einem 64-Bit-System aus. **7 pt**

There are 7 errors in the code below. Mark the incorrect lines with an X and correct the code. Assume a 64-bit system.

```
struct entry {
    struct entry *next;
};

struct intentry {
    int32_t data;
    struct entry e;
};
```

Solution:

```
struct intentry *entry_to_intentry(struct entry *e) {
    return (struct intentry *) ((char *) e - 8); /* mistake: e - 4 */
}

struct intentry *intentry_append(int32_t data, struct entry *e) {
    struct intentry *new = malloc(sizeof(*new));
    if (!new) exit(1); /* mistake: if (new) */
    new->data = data;
    new->e.next = e->next;
    e->next = &new->e; /* mistake: missing ->e */
    return new;
}

int32_t intentry_sum(struct entry *head) { /* mistake: void return type */
    int32_t sum = 0; /* mistake: missing initialization */
    for (struct entry *e = head->next; e; e = e->next) { /* mistake: skip head */
        sum += entry_to_intentry(e)->data;
    }
    return sum;
}

void intentry_free(struct entry *head) {
    struct entry *e = head->next;
    while (e) {
        struct intentry *ie = entry_to_intentry(e);
        e = e->next; /* mistake: use after free */
        free(ie);
    }
    head->next = NULL;
}

/* example usage on next page */
```

Welche Datenstruktur implementiert der Code? **0.5 pt**

Which data structure does the code implement?

Solution:

The code implements a linked list.

Was gibt das Programm (nach Korrektur aller Fehler) bei Ausführung der untenstehenden main-Funktion aus? **1 pt**

What does the program (after correcting all errors) print when executing the main function below?

```
int main() {
    struct entry head = {0};
    intentry_append(3, &head);
    intentry_append(2, &head);
    intentry_append(1, &head);
    printf("sum␣=␣%"PRId32"\n", intentry_sum(&head));
    intentry_free(&head);
}
```

Solution:

sum = 6

b) Die Funktion `f()` des untenstehenden C-Programms wird zu der Assembly rechts kompiliert.

The function `f()` in the C program below compiles to the assembly on the right.

main.c

```
int f(int x, int y) {
  int sum = x + y;
  return sum;
}

int main() {
  return f(5, 7);
}
```

main.S (Ausschnitt / *excerpt*)

```
f(int, int):
        push    ebp
        mov     ebp, esp
        sub     esp, 4
        mov     edx, DWORD PTR [ebp+8]
        mov     eax, DWORD PTR [ebp+12]
        add     eax, edx
        mov     DWORD PTR [ebp-4], eax
        mov     eax, DWORD PTR [ebp-4]
        leave
        ret
```

Welche Aufrufkonvention kommt hier zum Einsatz? **0.5 pt**

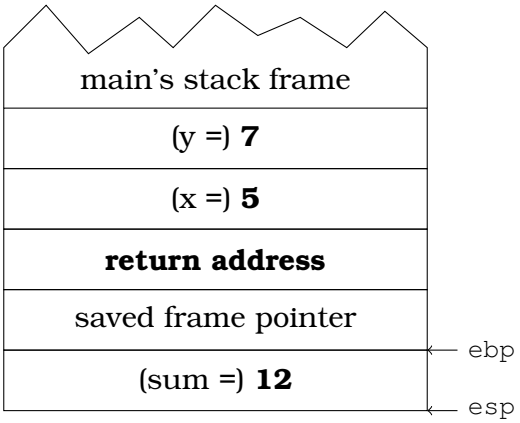
Which calling convention is used here?

Solution:

cdecl

Füllen Sie in dem untenstehenden Diagramm den Inhalt des Stacks vor Ausführung der `leave`-Instruktion in `f()` aus. Geben Sie Zahlenwerte an wo möglich, ansonsten eine Beschreibung des Werts. **2 pt**

In the diagram below, fill in the contents of the stack before execution of the `leave` instruction in `f()`. Write numeric values if possible, otherwise give a description of the value.



c) Die Funktion `strncat()` konkateniert zwei Strings. Implementieren Sie die Funktion `my_strncat()` entsprechend der Beschreibung in der Manpage zu `strncat()`. Rufen Sie dabei keine Funktionen auf. **4 pt**

The function `strncat()` concatenates two strings. Without calling any functions, implement the function `my_strncat()` according to the manpage for `strncat()`.

Solution:

```
char *my_strncat(char *dest, const char *src, size_t n) {
    char *ret = dest;
    while (*dest) dest++;

    while (*src && n-- > 0)
        *dest++ = *src++;
    *dest = '\0';

    return ret;
}

char *my_strncat_alt(char *dest, const char *src, size_t n) {
    size_t dest_len = 0;
    for (char *d = dest; *d; d++) dest_len++;

    size_t i;
    for (i = 0; i < n && src[i] != '\0'; i++)
        dest[dest_len + i] = src[i];
    dest[dest_len + i] = '\0';

    return dest;
}
```

(1 P) *skip to end or calculate length of dest,* **(1 P)** *decrement and check n,* **(1 P)** *copy from src to dest,* **(0.5 P)** *0-terminate dest,* **(0.5 P)** *return*

Total: 15.0pt

Aufgabe P2: Dateikomprimierung

Assignment P2: File Compression

Sie sollen ein Programm `compress` schreiben, das die Dateien, die durch die Kommandozeilenparameter spezifiziert werden, parallel einliest, komprimiert, und dann die Dateiinhalte durch die jeweiligen komprimierten Daten ersetzt. So soll zum Beispiel `compress A B` die Inhalte der Dateien `A` und `B` durch ihre komprimierte Form ersetzen.

- Sie müssen keine C-Header inkludieren.
- Sie müssen keine Fehlerbehandlung implementieren.
- Geben Sie jegliche im Code angeforderten Ressourcen wieder frei.

You shall write a program `compress` which reads the files specified by the command line parameters in parallel, compresses them, and then replaces the file contents with the corresponding compressed data. For example, `compress A B` replaces the contents of the files `A` and `B` by their compressed form.

- *You do not need to include any C headers.*
- *You do not have to implement any error handling.*
- *Free all resources allocated in the code.*

```
/* Global variables */
pthread_mutex_t mutex; /* mutex which protects "count" and "files" */
size_t count;          /* number of files yet to be processed */
char **files;          /* array of files yet to be processed */
                       /* ("files[0]" to "files[count-1]" are valid) */
```

a) Die folgende Funktion `replace_file()` ersetzt jegliche Inhalte der existierenden Datei `path` durch die Daten der Länge `size` an der Adresse `data`. Vervollständigen Sie den zweiten Parameter für `open()`, damit eine solche Ersetzung stattfindet. **0.5 pt**

The following function `replace_file()` replaces any content of the existing file `path` by the data of size `size` at the address `data`. Complete the second parameter of `open()` so that such a replacement occurs.

Solution:

```
void replace_file(const char *path, const char *data, size_t size) {
    int fd = open(path, O_WRONLY | O_TRUNC);
    write(fd, data, size);
    close(fd);
}
```

The file needs to be opened for writing (`O_WRONLY`) and needs to be truncated to remove old content (`O_TRUNC`) **(0.5 P)**.

Vervollständigen Sie die Funktion `read_file()`, die die Inhalte der Datei `path` in den Speicher liest und einen Pointer auf die Daten zurückgibt. Die Länge der Daten soll an die durch `size` spezifizierte Adresse geschrieben werden. **5 pt**

- Bestimmen Sie vor dem Einlesen die Größe der Datei, um Speicher zu allozieren.
- Achtung: Der Speicher muss gegebenenfalls in einer der folgenden Teilaufgaben freigegeben werden.

Complete the function `read_file()` which reads the contents of the file `path` into memory and returns a pointer to the data. The size of the data shall be written to the address specified by `size`.

- *Determine the file size before reading the data to allocate memory.*
- *Caution: The memory may have to be released in one of the following sub-tasks.*

Solution:

```
char *read_file(const char *path, size_t *size) {
    char *data;
    struct stat s;
    int fd;

    stat(path, &s);
    *size = s.st_size;

    fd = open(path, O_RDONLY);
    data = malloc(*size);
    read(fd, data, *size);
    close(fd);

    return data;
}
```

Open the file **(0.5 P)** *for reading* **(0.5 P)**, *call `stat()` or `fstat()` to determine the size of the file* **(1.5 P)**, *write the size to `*size`* **(0.5 P)**, *allocate a buffer* **(0.5 P)**, *read the data* **(1.0 P)**, *close the file* **(0.5 P)**.

b) Vervollständigen Sie die Funktion `compress_file()`, die eine einzige Datei (`path`) einliest, komprimiert und die Dateiinhalte mit den komprimierten Daten ersetzt. **3 pt**

- Verwenden Sie `compressBound()` und `compress()` aus der Zlib-Library zum Komprimieren. Hierfür finden Sie eine Manpage im Anhang der Klausur.
- Beachten Sie die Hinweise der Manpage zum Speicherbedarf der komprimierten Daten.

Complete the function `compress_file()` which reads a single file (`path`), compresses the data, and replaces the file contents with the compressed form.

- *For compression, use `compressBound()` and `compress()` from the Zlib library. A manpage for both can be found appended to the exam.*
- *Note the hints from the manpage with regards to memory usage for compressed data.*

Solution:

```
void compress_file(const char *path) {
    size_t in_size;
    unsigned long out_size;
    char *input, *output;

    input = read_file(path, &in_size);

    out_size = compressBound(in_size);
    output = malloc(out_size);
    /*  zlib expects unsigned char pointers, char* is usually signed */
    compress((unsigned char*)output, &out_size,
             (const unsigned char*)input, in_size);

    replace_file(path, output, out_size);

    free(input);
    free(output);
}
```

Calculate the maximum size of the compressed data **(0.5 P)**, *allocate a buffer* **(0.5 P)**, *call* `compress()` **(0.5 P)** *with the correct parameters* **(1.0 P)**, *and free the buffers again* **(0.5 P)**.

c) Das Programm soll mittels Threads parallelisiert werden, die jeweils die folgende Funktion `thread_entry()` ausführen. Vervollständigen Sie diese Funktion, die in einer Schleife jeweils einen Eintrag aus der durch `files` und `count` spezifizierten Liste entnimmt und die entsprechende Datei mittels `compress_file()` komprimiert. Dies soll wiederholt werden, solange noch Einträge vorhanden sind. **4 pt**

- Zugriffe auf die globalen Variablen müssen durch `mutex` geschützt werden. Komprimieren soll parallel erfolgen können.
- Sie können das erste Element aus der Liste entfernen, indem Sie die globale Variable `files` auf das zweite Element der Liste zeigen lassen.

The program shall be parallelized using threads which all call the following function `thread_entry()`. *Complete this function which in a loop removes an entry from the list specified by* `files` *and* `count` *and compresses the corresponding file via* `compress_file()`. *This shall be repeated as long as entries remain.*

- *Access to the global variables needs to be protected using* `mutex`. *Parallel compression shall be possible.*
- *You can remove the first element from the list by letting the global variable* `files` *point to the second element of the list.*

Solution:

```
void *thread_entry(void *param) {
    const char *path;
    (void)param; /* unused */

    while (1) {
        pthread_mutex_lock(&mutex);
        if (count == 0) {
```

```
            path = NULL;
        } else {
            path = files[0];
            files = files + 1;
            count = count - 1;
        }
        pthread_mutex_unlock(&mutex);

        if (path == NULL) {
            break;
        }
        compress_file(path);
    }

    return NULL;
}
```

Loop **(0.5 P)** *until all files have been processed* **(1.0 P)**, *use the mutex to protect access to the global variables* **(1.0 P)**, *remove the first list entry* **(1.0 P)** *and call* `compress_file()` *for the corresponding file* **(0.5 P)**. *Note that* `compress_file()` *must not be in the critical section, as then parallel compression would not be possible.*

d) Vervollständigen Sie die `main()`-Funktion des Programms, die eine feste Anzahl an Threads erstellt, die `thread_entry()` aufrufen, um die Dateien in `files` zu komprimieren. **2.5 pt**

Complete the `main()` *function of the program which creates a fixed number of threads which call* `thread_entry()` *to compress the files specified by* `files`.

Solution:

```
#define NUM_THREADS 8
int main(int argc, char **argv) {
    int i;
    pthread_t threads[NUM_THREADS];

    /* initialization */
    files = argv + 1;
    count = argc - 1;
    pthread_mutex_init(&mutex, NULL);

    /* create threads */
    for (i = 0; i < NUM_THREADS; i++) {
        pthread_create(&threads[i], NULL, thread_entry, NULL); /* 1.5P */
    }

    /* wait for the threads to exit */
    for (i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    return 0;
}
```

Call `pthread_create()` **(0.5 P)** *with the correct arguments* **(1.0 P)** *to create threads and call* `pthread_join()` **(0.5 P)** *with the correct arguments* **(0.5 P)** *to wait for their completion.*

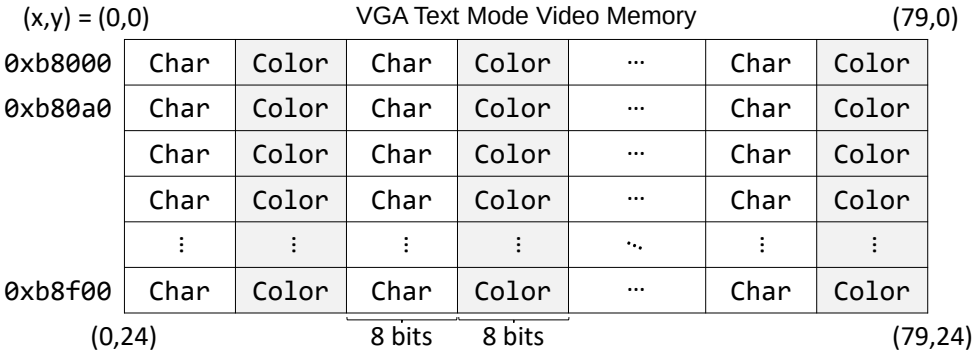
Total: 15.0pt

Aufgabe P3: VGA Terminal

Assignment P3: VGA Terminal

Der Video Graphics Array (VGA) Standard definiert eine einfache Schnittstelle, um Ausgaben auf dem Bildschirm zu erzeugen. VGA unterstützt dabei einen Textmodus mit 80×25 Zeichen (Breite×Höhe), bei dem ASCII-Zeichen angezeigt werden können, indem sie in einen vordefinierten Speicherbereich ab Adresse `0xb8000` geschrieben werden. Jedes Zeichen umfasst 16 Bits, wobei das erste Byte das ASCII-Zeichen und das zweite Byte eine Farbkodierung enthält (unabhängig von der Endianness der CPU).

The Video Graphics Array (VGA) standard defines a simple interface to produce output on the screen. VGA supports a text mode with 80×25 characters (width×height), which allows displaying ASCII characters by writing them to a predefined memory area starting at address `0xb8000`. Each character is 16 bits, where the first byte contains the ASCII code and the second byte contains a color code (independent of the endianness of the CPU).




```c
#include <inttypes.h>

#define VGA_W    80          /* Screen width in characters              */
#define VGA_H    25          /* Screen height in characters             */

#define VGA_MEM  0xb8000     /* Base address of VGA memory              */

uint8_t term_fg, term_bg;    /* Current terminal fore-/background color */
uint8_t term_x,  term_y;     /* Current terminal cursor position        */

/* VGA word (character + color) declaration */
typedef struct {
    uint8_t chr;             /* ASCII character                         */
    uint8_t color;           /* Color code                              */
} vga_word_t;

/* Pointer to VGA memory */
vga_word_t *vga = (vga_word_t*)(VGA_MEM);

/* VGA access macro */
#define VGA(x, y) (vga[((y) * VGA_W) + (x)])
```

a) Ergänzen Sie auf Seite 20 einen Typ `vga_word_t`, der ein Zeichen im VGA-Speicher inkl. Farbcodierung repräsentiert (d. h. ein VGA-Wort). Der Typ soll unabhängig von der Endianness des Zielsystems sein. **1 pt**

On page 20, add a type `vga_word_t` which represents a character in VGA memory including its color code (i.e., one VGA word). The type shall be independent from the endianness of the target system.

Solution:

(See global variables)

Declare struct and typedef **(0.5 P)**, *declare members (right type, right order)* **(0.5 P)**

b) Vervollständigen Sie die Funktion `vga_word()`, welche für ein ASCII-Zeichen (`c`) und eine Vorder- (`fg`) und Hintergrundfarbe (`bg`) ein `vga_word_t` zurückgibt. **2 pt**

- Die Farben werden im VGA-Speicher als 4-bit Indizes in eine VGA-Farbpalette angegeben. Die Vordergrundfarbe ist in den niederwertigen 4 Bits und die Hintergrundfarbe in den höherwertigen 4 Bits des `Color` Bytes gespeichert.
- Übernehmen Sie jeweils nur die niederwertigsten 4 Bits von `fg` und `bg` in das VGA-Wort.

Complete the function `vga_word()` which returns a `vga_word_t` for a given ASCII character (`c`), foreground color (`fg`), and background color (`bg`).

- *The colors are specified in VGA memory as 4-bit indices into a VGA color palette. The foreground color is stored in the least significant bits and the background color in the most significant bits of the `Color` byte.*
- *Transfer only the least significant 4 bits of `fg` and `bg` to the VGA word.*

Solution:

```
vga_word_t vga_word(char c, uint8_t fg, uint8_t bg)
{
    vga_word_t r;

    r.chr = (uint8_t)c;
    r.color = ((bg & 0xf) << 4) | (fg & 0xf);

    return r;
}
```

Set `c` **(0.5 P)**, *set `fg` and `bg` with shift, and apply masks (for `bg` optional)* **(1.0 P)**, *return* **(0.5 P)**

c) Vervollständigen Sie das Makro `VGA(x, y)` auf Seite 20, welches mittels der Variablen `vga` das VGA-Wort an der Position (`x`,`y`) adressiert. Das Makro soll Lese- und Schreibzugriff der folgenden Form ermöglichen, wobei `x` und `y` beliebige Ausdrücke sein dürfen: **1 pt**

Complete the macro `VGA(x, y)` on page 20 which addresses the VGA word at position (`x`,`y`) using the variable `vga`. The macro shall allow read and write access of the following form, where `x` and `y` may be arbitrary expressions:

```
vga_word_t w;

w = VGA(x1, y1);
VGA(x2, y2) = w;
```

Solution:

(See global variables)

Correct index computation **(0.5 P)**, *correct reference and use of parentheses* **(0.5 P)**

d) Vervollständigen Sie die Funktion `term_clear()`, welche den vollständigen VGA-Speicher mit Leerzeichen (`' '`) in der durch `fg` und `bg` angegebenen Farbkodierung überschreibt. **2.5 pt**

- Übernehmen Sie die Werte von `fg` und `bg` als neue Terminal-Farbkodierung.
- Setzen Sie die Position des Cursors auf (0,0) zurück.

Complete the function `term_clear()` which overwrites the entire VGA memory with spaces (`' '`) in the color code specified by `fg` and `bg`.

- *Take the values of `fg` and `bg` as new terminal color code.*
- *Reset the position of the cursor to (0,0).*

Solution:

```
void term_clear(uint8_t fg, uint8_t bg)
{
    term_fg = fg;
    term_bg = bg;

    term_x = 0;
    term_y = 0;

    vga_word_t word = vga_word('␣', fg, bg);
    for (int y = 0; y < VGA_H; y++) {
        for (int x = 0; x < VGA_W; x++) {
            VGA(x, y) = word;
        }
    }
}
```

Set `term_X` variables **(0.5 P)**, *nested loop* **(1.0 P)**, *setting current VGA word* **(1.0 P)**

e) Vervollständigen Sie die Funktion `term_scroll()`, welche den Inhalt des VGA-Speichers um eine Zeile nach oben rückt und dabei die oberste Zeile verwirft. **3 pt**

- Füllen Sie die neue Zeile mit Leerzeichen (`' '`) in der Terminal-Farbkodierung.
- Wenn möglich, rücken Sie den Cursor ebenfalls um eine Zeile nach oben.

Complete the function `term_scroll()` which moves the contents of the VGA memory up one line, discarding the top line.

- *Fill the new line with spaces (`' '`) in the terminal color code.*
- *If possible, also move the cursor up one line.*

Solution:

```
void term_scroll(void)
{
    for (int y = 1; y < VGA_H; y++) {
        /* Alternative arguments:
         *  dest: vga + (y - 1) * VGA_W
         *  src : vga + y * VGA_W
         */

        memcpy(&VGA(0, y - 1), &VGA(0, y),
               VGA_W * sizeof(vga_word_t));
    }

    /* Alternative (memmove allows overlapping dest and src):
     * memmove(vga, vga + VGA_W,
     *         VGA_W * (VGA_H - 1) * sizeof(vga_word_t));
     */

    vga_word_t word = vga_word('␣', term_fg, term_bg);
    for (int x = 0; x < VGA_W; x++) {
        VGA(x, VGA_H - 1) = word;
    }

    if (term_y > 0) {
        term_y--;
    }
}
```

Move VGA memory **(1.5 P)**, *clear last line* **(1.0 P)**, *update cursor* **(0.5 P)**

f) Vervollständigen Sie die Funktion `term_putc()`, welche das ASCII-Zeichen in der Terminal-Farbkodierung an der aktuellen Position des Cursors ausgibt. **5.5 pt**

- Geben Sie nur druckbare ASCII-Zeichen aus (siehe ASCII-Tabelle).
- Implementieren Sie die Steuerzeichen für Wagenrücklauf (CR - Cursor zum Zeilenanfang) und Zeilenvorschub (LF - Cursor um eine Zeile nach unten).
- Am Zeilenende springt der Cursor an den Anfang der nächsten Zeile.
- Rufen Sie `term_scroll()` auf, wenn nötig.

Complete the function `term_putc()` *which outputs the ASCII character in the terminal's color code at the current position of the cursor.*

- *Only output printable ASCII characters (see ASCII table).*
- *Implement the control code for carriage return (CR - cursor to beginning of line) and line feed (LF - cursor down one line).*
- *At the end of the line the cursor jumps to the beginning of the next line.*
- *Call* `term_scroll()` *if necessary.*

Solution:

```
void term_putc(char c)
{
        if ((c >= 0x20) && (c <= 0x7e)) {
            VGA(term_x, term_y) = vga_word(c, term_fg, term_bg);
            term_x++;
        } else if (c == '\r') {
            term_x = 0;
        } else if (c == '\n') {
            term_y++;
        }

        if (term_x >= VGA_W) {
            term_x = 0;
            term_y++;
        }

        if (term_y >= VGA_H) {
            term_scroll();    /* Adjusts term_y */
        }
}
```

Check for printable ASCII char **(1.0 P)**, *output of ASCII char and update of cursor* **(1.0 P)**, *handling of control codes* **(1.5 P)** *move cursor to next line* **(1.0 P)** *scroll text* **(1.0 P)**

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	␣	!	"	#	$	%	&	’	(	)	*	+	,	−	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	∼	DEL

Total: 15.0pt