Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

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

Musterlösung Hauptklausur
03.03.2021

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 23 Blättern: Einem Deckblatt und 22 Aufgabenblättern mit insgesamt 5 Theorieaufgaben, 3 Programmieraufgaben und 0 Seiten Man-Pages.

  The examination consists of 23 pages: One cover sheet and 22 sheets containing 5 theory assignments, 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) Nennen Sie je ein Beispiel für einen Mechanismus und für eine Policy eines Betriebssystems. **1 pt**

Name one example for a mechanism as well as for a policy of an operating system.

Solution:

- *Examples for mechanisms* **(0.5 P)***: Dispatcher/threads, swap file, . . .*
- *Examples for policies* **(0.5 P)***: Scheduler, page replacement policy, . . .*

Welchen Vorteil bietet die Trennung zwischen Mechanismus und Policy? **1 pt**

Which advantage does the separation between mechanism and policy provide?

Solution:

More flexibility: Replacing the policy does not require rewriting the mechanism.

b) Erklären Sie den Unterschied zwischen Nebenläufigkeit und Parallelität. **1 pt**

Explain the difference between concurrency and parallelism.

Solution:

Concurrency (pseudo-parallelism) means that the operating system implements context switching between tasks on a single CPU **(0.5 P)***, whereas parallelism means that multiple tasks run on multiple CPUs at the same time* **(0.5 P)***.*

In welchen Situationen erhöhen Nebenläufigkeit bzw. Parallelität jeweils die Performance des Systems? **1 pt**

In which situations do concurrency and parallelism each increase the performance of the system?

Solution:

Concurrency allows other tasks to run while one task waits for I/O – the overlap between CPU and I/O improves performance **(0.5 P)***. Parallelism increases performance whenever multiple tasks are runnable at the same time* **(0.5 P)***.*

c) Java wird allgemein als einfachere Programmiersprache als C angesehen. Erläutern Sie einen Grund, weshalb es verglichen mit C jedoch schwieriger ist, einen Betriebssystemkernel in Java zu schreiben. **1 pt**

Java is commonly viewed as an easier programming language than C. Explain one reason why it is, however, more difficult to write an operating system kernel in Java compared to C.

Solution:

Potential answers:

- *Java uses memory management based on garbage collection – the garbage collector itself adds to the work required to program a kernel and cannot be written in Java alone* **(1.0 P)**.
- *Java does not allow access to arbitrary memory, although the OS frequently needs such access to arbitrary addresses to access device registers or to perform memory management* **(1.0 P)**.

d) Erklären Sie, was man unter dem Begriff *Limited Direct Execution* versteht. **1 pt**

Explain what is meant by the term limited direct execution.

Solution:

Limited direct execution means that user-space code is allowed to perform most operations directly on the CPU without OS intervention **(0.5 P)**, *and only those operations which affect the safety or stability of the system are reserved to the OS* **(0.5 P)**.

Microkernel-Betriebssysteme implementieren Gerätetreiber und Komponenten des Betriebssystems als separate Prozesse im User-Space, anstatt sie im gemeinsamen Kernel-Space zu platzieren. Nennen Sie einen Vorteil und einen Nachteil dieser Architektur. Begründen Sie jeweils Ihre Antworten. **3 pt**

Microkernel operating systems implement device drivers and operating system components as separate processes in user space instead of placing them in the shared kernel space. Name one advantage and one disadvantage of this architecture. Justify your answers.

Solution:

Advantages:

- *Bugs in drivers have less impact on the safety/stability of the whole system* **(0.5 P)**. *For example, accesses to invalid pointers do not have the potential to crash the OS kernel* **(1.0 P)**.

Disadvantages:

- *The architecture causes increased overhead compared to traditional operating systems* **(0.5 P)**. *As the system is partitioned with finer granularity, more system calls and IPC operations are required to communicate between drivers and the kernel* **(1.0 P)**.

Total: 9.0pt

Aufgabe T2: Prozesse und Threads

Assignment T2: Processes and Threads

a) Wo befinden sich die Prozesskontrollblöcke (PCB) im Adressraum? **0.5 pt**

Where are the process control blocks (PCB) located in the address space?

Solution:

The PCBs are stored in the kernel's memory range.

Note: *The kernel memory can usually be located anywhere in the virtual address space, so answers like "at high/low addresses" are not specific enough.*

Kann ein Debugger in einem Mehrprozessorsystem die Ausführung eines gleichzeitig laufenden Prozesses beobachten, indem er dessen Zustand aus PCB und TCB liest? Erklären Sie. **1 pt**

Is it possible for a debugger in a multiprocessor system to observe the execution of another process running in parallel by reading its PCB and TCB? Explain.

Solution:

No, although the PCB and TCB contain the execution state of a process, most information is only valid while the process is not running.

Note: **(0.5 P)** *for answers like "no, the debugger cannot access kernel memory" (it can, usually via special system calls such as ptrace).*

b) Geben Sie zwei freiwillige Ereignisse an, die einen Threadwechsel im One-to-One-Modell auslösen können. **1 pt**

Give two voluntary events that might trigger a thread switch in the One-to-One model.

Solution:

- *calling* `yield()`
- *executing a blocking system call*
- *waiting on a mutex or semaphore*
- *thread exit*

c) Was ist die Aufgabe eines Langzeitschedulers? **0.5 pt**

What does a long-term scheduler do?

Solution:

The long-term scheduler selects which processes to put into the ready queue.

d) Welches Problem tritt häufig beim Einsatz von striktem Prioritätsscheduling auf? Beschreiben Sie das Problem kurz. **1 pt**

Which issue commonly occurs in systems with strict priority scheduling? Describe the issue shortly.

Solution:

Systems with strict priority scheduling often suffer from ***starvation*** **(0.5 P)***: Low-priority processes will never run if there is a runnable higher-priority process for an extended period of time.* **(0.5 P)**

(alternative solution: priority inversion)

e) Gegeben seien drei Prozesse auf einem Einprozessorsystem mit einem Multi-Level Feedback Queue Scheduler. **5 pt**

- Der Scheduler verwendet die drei untenstehenden Warteschlangen.
- Jeder Prozess startet in der Warteschlange 2 und blockiert einmal, nachdem er eine gewisse Dauer gelaufen ist.
- Ein Prozess, der seine Zeitscheibe komplett ausnutzt, wird in die nächste Warteschlange mit niedrigerer Priorität verschoben. Ansonsten wird er in die Warteschlange mit nächsthöherer Priorität verschoben.
- Der Scheduler wird nach jeder halben Zeiteinheit ausgeführt.
- Das Ablaufdiagramm gibt an, welcher Prozess wann ausgeführt wird (Zeile P) und in welche Warteschlange er als nächstes eingefügt wird (Zeile Q).

Füllen Sie die fehlenden Informationen in den beiden untenstehenden Tabellen aus. Markieren Sie zusätzlich zu Ihrer Lösung eine Zelle mit (?), falls mehrere Lösungen möglich sind.

Consider three processes on a uniprocessor system with a Multi-Level Feedback Queue scheduler.

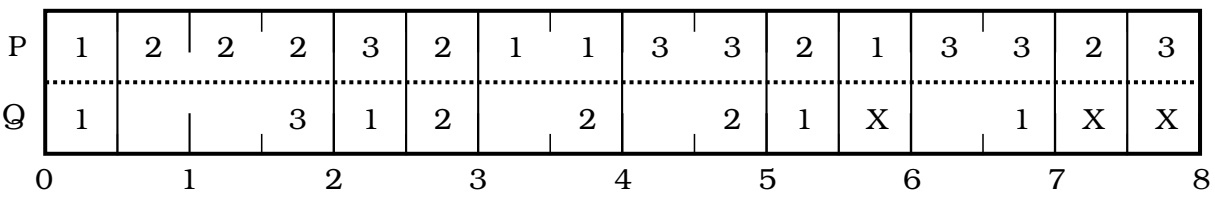
- *The scheduler uses the three queues given below.*
- *Each process starts in queue 2 and blocks once after running for a certain time.*
- *A process that uses all of its timeslice is inserted into the next queue with lower priority. Otherwise, it is inserted into the next queue with higher priority.*
- *The scheduler is executed after each half unit of time.*
- *The scheduling plan shows which process runs in each time slot (row P), as well as the queue the process will be inserted into next (row Q).*

Fill in the missing information in the two tables below. In addition to your solution, mark a cell with (?) if multiple solutions are possible.

Queue	Scheduler	Timeslice	Priority
1	Round Robin	*1*	High
2	Round Robin	*1.5*	Mid
3	Round Robin	2	Low

Process	Arrival Time	Job Length	Blocks after ...	Blocks for ...
1	0	2	*0.5*	*2.5*
2	0.1	*3*	*2.5*	*1.5*
3	1	*3*	*0.5*	*1 / 1.5 (?)*

Ablaufplan/*scheduling plan*:



All job lengths **(0.5 P)**, every other cell **(0.5 P)**, correct ambiguous cell **(0.5 P)** **Total: 9.0pt**

Aufgabe T3: Speicher

Assignment T3: Memory

a) Gehen Sie von einem System mit 16-Bit virtuellen Adressen aus, welches vier Segmente unterstützt und die Segmentnummer in den höchstwertigsten Bits (MSB) der virtuellen Adresse kodiert. Verwenden Sie die gegebene Segmenttabelle, um die Lücken in den Übersetzungen zu füllen. **3 pt**

Assume a system with 16-bit virtual addresses that supports four segments and encodes the segment number in the most significant bits (MSB) of the virtual address. Use the provided segment table to fill the gaps in the translations.

Segment Nr.	Base	Limit
0	0x1000	0x3000
1	0x4000	0x0A00
2	0x8400	0x1000
3	0xEE00	0x0200

Solution:

(0.5 P) *for the first correct cell per row and* **(1 P)** *if a row is entirely correct. Underlined values were provided as part of the question.*

Virtual Address	*Segment Nr.*	*Offset*	*Physical Address*	
0x8450	*2*	*0x0450*	*0x8400+0x0450 = 0x8850*	**(1 P)**
0x40A0	*1*	*0x00A0*	*0x4000+0x00A0 = 0x40A0*	**(1 P)**
0xC1DE	*3*	*0x01DE*	*0xEFDE*	**(1 P)**

b) Nennen Sie zwei Nachteile von segmentbasierter Speicherverwaltung gegenüber seitenbasierter Speicherverwaltung. **1 pt**

Give two disadvantages of segment-based memory management compared to page-based memory management.

Solution:

- *Segments need to be kept contiguous in physical memory.*
- *Segments can only be swapped in and out as a whole.*
- *Segmentation may suffer from external fragmentation.*

c) Wie groß ist der virtuelle Adressraum für eine 5-stufige Seitentabelle mit 64-Bit Einträgen und 4 KiB Seiten? Machen Sie Ihren Rechenweg deutlich. **1 pt**

What size is the virtual address space for a 5-level page table with 64-bit entries and 4 KiB pages? Explain your calculation.

Solution:

$2^{12}/2^3 = 2^9$ *page table entries per 4 KiB page.*
$(2^9)^5 = 2^{45}$ *pages in the address space á* 2^{12} *bytes:* 2^{57} *bytes = 128 PiB*

d) Eine bereits ausgewählte Heap-Seite soll aus dem Adressraum von Prozess A ausgelagert und dem Prozess B als freie Seite zur Verfügung gestellt werden. Erläutern Sie, welche grundlegenden Schritte für diesen Vorgang notwendig sind. Gehen Sie davon aus, dass der Kernel über ein eigenes Mapping der physischen Seite verfügt. **2.5 pt**

An already selected heap page should be swapped out from the address space of process A and made available to process B as a free page. Explain which basic steps are necessary for this operation. Assume that the kernel has its own mapping of the physical page.

Solution:

(1) Invalidate user mapping in process A **(0.5 P)** *(unset P-bit in corresponding PTE)*

(2) Flush TLB entry **(0.5 P)**

(3) Write page to swap file **(0.5 P)** *via kernel mapping. Note offset somewhere.*

(4) Clear page **(0.5 P)** *via kernel mapping*

(5) Create new user mapping in process B **(0.5 P)** *(configure PTE with correct frame number and set P-bit)*

(-0.5 P) *for the wrong order.*

e) Was versteht man unter dem Begriff *Thrashing*? **0.5 pt**

What is meant by the term thrashing*?*

Solution:

The system is under heavy memory pressure and busy swapping pages in and out. CPU utilization is low as processes wait for pages to be fetched from secondary storage.

Weshalb kann die Berechnung von Working Sets in dieser Situation nützlich sein? Erläutern Sie Ihre Antwort. **1 pt**

Why can the calculation of working sets be useful in this situation? Justify your answer.

Solution:

The working set comprises all pages that are currently needed by a process to make progress. In situations of high memory pressure, this information can drive page replacement decisions so that less recently accessed pages are swapped out first. In critical situations, the working set can be used to select processes for suspension.

Total: 9.0pt

Aufgabe T4: Koordination und Kommunikation von Prozessen

Assignment T4: Process Coordination and Communication

a) Eingehende Nachrichten sollen von einer Gruppe von Prozessen bearbeiten werden können. Welche Art des Message Passing müssen Sie verwenden? **0.5 pt**

Incoming messages shall be processed by a group of processes. Which type of message passing do you have to use?

Solution:

Indirect message passing (or message passing with mailboxes)

b) Nennen Sie die drei Klassen von Gegenmaßnahmen gegen Deadlocks. **1.5 pt**

Name the three classes of countermeasures against deadlocks.

Solution:

- *Deadlock prevention* **(0.5 P)**
- *Deadlock avoidance* **(0.5 P)**
- *Deadlock detection/recovery* **(0.5 P)**

c) Erklären Sie, weshalb ein Spinlock nicht dafür geeignet ist, Zugriffe von zwei Threads zu synchronisieren, die auf dem gleichen CPU-Kern ausgeführt werden. **1.5 pt**

Explain why a spinlock is not suitable for synchronization of accesses by two threads which execute on the same CPU core.

Solution:

The thread spinning can potentially preempt the thread holding the lock **(1.0 P)**. *At this point, no progress can be made until the thread holding the lock is scheduled again, yet the thread which is spinning wastes CPU resources* **(0.5 P)**.

Kernel verwenden häufig Spinlocks, Anwendungen dagegen nicht. Weshalb ist ein sinnvoller Einsatz im Kernel einfacher als in Anwendungen? **1 pt**

Kernels often use spinlocks, whereas applications do not. Why is sensible use easier in the kernel than in applications?

Solution:

The kernel can disable preemption while holding a spinlock by disabling interrupts. Disabling preemption solves the problem described in the last question **(1.0 P)**.

Solutions arguing that critical sections in the kernel are shorter than in user space are incorrect – often, critical sections are long (e.g., in the file system due to I/O latency), the resulting problems hurt more as they affect the whole system, and preemption can cause long waiting times even for short critical sections.

Erklären Sie, wie ein *Two-Phase-Lock* funktioniert. **1 pt**

Explain how a two-phase lock *works.*

Solution:

The lock first spins for a fixed amount of time **(0.5 P)**. *If the lock cannot be acquired within this time* **(0.5 P)**, *the threads executes a syscall to sleep until the lock is available* **(0.5 P)**.

d) Beschreiben Sie die Auswirkung einer `fence`-Instruktion auf das Verhalten des Prozessors. Weshalb ist dieses Verhalten für den Code zum Betreten bzw. Verlassen eines kritischen Abschnittes notwendig? **2 pt**

Describe the impact of a `fence` instruction on the behavior of the processor. Why is this behavior necessary for the code to enter and exit a critical section?

Solution:

The instruction restricts the reordering of other instructions around the fence instructions by the processor **(1.0 P)**. *Instructions must not be reordered at the boundaries of a critical section, or else parts of the code in the critical section may actually take effect while no lock is held* **(1.0 P)**

Note that the implementation of a spinlock itself uses an atomic instruction, not a fence.

e) Skizzieren Sie eine Methode, mit der das Betriebssystem die Kommunikation zwischen zwei beliebigen mittels Shared Memory kommunizierenden Prozessen aufzeichnen kann. **1.5 pt**

Sketch a method with which the operating system can record communication between two arbitrary processes which communicate via shared memory.

Solution:

Potential solutions:

- *The shared memory region is unmapped from the processes* **(0.5 P)**. *Each access generates a page fault, the OS emulates and records the accesses* **(1.0 P)**.
- *The shared memory region is unmapped from the processes* **(0.5 P)**. *When one process accesses the shared memory region, it is mapped for this process and unmapped from the other. During each such switch, the contents of the region are recorded* **(1.0 P)**.

Solutions based on frequent polling of the contents of the memory region are incorrect due to the resulting race conditions and are awarded **(0.5 P)**. *Solutions valid only for modified applications (e.g., if message passing is implemented via specific system calls that can be intercepted by the OS) are incorrect as the question asks about* arbitrary processes, *such solutions are also awarded only* **(0.5 P)**. **Total: 9.0pt**

Aufgabe T5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Nennen Sie eine Blockallokationsstrategie und bewerten Sie ihre Eignung für wahlfreien Dateizugriff. Erklären Sie, welche Schritte für den Dateizugriff bei einem bestimmten Offset notwendig sind. **2.5 pt**

Name a block allocation policy and rate its suitability for random file access. Explain the necessary steps for accessing file contents at a certain offset.

Solution:

Contiguous allocation *Random access is very efficient. Obtain start block from FAT, check offset against file length, calculate target block from start block and offset, read target block.*

Chained allocation *Random access is very inefficient. Obtain start block from FAT, repeatedly read data block from disk and follow pointer to the next block until the target block is reached.*

Linked List Allocation/FAT *Random access is slightly more efficient than with chained allocation because the block chain is stored in RAM (otherwise, similar description of steps).*

Indexed Allocation *Random access is slightly less efficient than with contiguous allocation. Read inode for file, follow pointers to the correct indirect block for the requested offset, read target block.*

name **(0.5 P)**, *rating* **(0.5 P)**, *steps* **(1.5 P)**

b) Sie schreiben einen vollständigen Block auf einem RAID 4-System mit vier Festplatten für Daten und einer Festplatte für Parität. Wie viele Schreib- bzw. Lesezugriffe in Blockgröße werden auf welchen Festplatten durchgeführt? **1 pt**

You are writing a full block to a RAID 4 system with four disks for data and one disk for parity. How many block-sized read and write accesses are performed on which of the disks?

Solution:

- *One read access to one of the data disks to obtain the old data in the block. We need this information to calculate the new parity.*
- *One read and one write access to the parity disk to update the parity for that block.*
- *One write access to one of the data disks to update the block.*

Note: *Although we could in theory read the corresponding data block from all other disks to calculate the parity, doing so would be very inefficient.*

Welches Problem hat RAID 4? Wie wird dieses Problem von RAID 5 gelöst? **1 pt**

What issue does RAID 4 have? How does RAID 5 solve this issue?

Solution:

The parity disk is accessed for every write on any disk and thus tends to fail quickly **(0.5 P)**. *With RAID 5, the parity is distributed across all disks* **(0.5 P)**.

Note: *A RAID 5 system cannot tolerate more disk failures than a RAID 4 system.*

c) Das untenstehende Shellskript wird auf einem Linux-System ausgeführt. Welchen Dateiinhalt haben die einzelnen Dateien danach? Füllen Sie die Tabelle aus. Schreiben Sie „X“, falls auf eine Datei nicht zugegriffen werden kann. **2.5 pt**

The shell script below is run on a Linux system. What content does each file have afterwards? Fill in the table. Write "X" if a file cannot be accessed.

```
# -n: no newline
echo -n A > a
echo -n B > b
# ln target link_name
ln a c
# -s: symlink
ln -s c d
ln -s b e
# >>: append to file
echo -n 1 >> c
echo -n 2 >> d
# mv source dest
mv b c
echo -n 3 >> d
```

File	Contents (`cat File`)
a	A12
b	**X**
c	B3
d	B3
e	**X**

d) Welche Betriebssystemkomponente erlaubt das Einhängen von mehreren Dateisystemen in einem gemeinsamen Ordnerbaum? **0.5 pt**

Which operating system component allows mounting multiple file systems into a shared directory tree?

Solution:

The Virtual File System (VFS).

e) Welcher Systemaufruf sorgt dafür, dass alle veränderten Blöcke im Dateisystemcache auf das Speichergerät geschrieben werden? **0.5 pt**

Which system call flushes all dirty blocks in the file system cache to the disk?

Solution:

one of `sync()`, `fsync()`, `msync()`

f) Beschreiben Sie einen Vorteil von Log-Structured-Dateisystemen (bzw. Copy-on-Write-Dateisystemen) gegenüber Journaling-Dateisystemen. **1 pt**

Describe an advantage of log-structured file systems (or copy-on-write file systems) over journaling file systems.

Solution:

- *Journaling file systems need to write all data twice: once to the journal and a second time to the actual data blocks. A copy-on-write file system can provide the same crash consistency guarantees without the second data write.*
- *Copy-on-write file systems can provide complex features such as snapshots or block-level data deduplication with little overhead compared to journaling file systems.*
- *Log-structured file systems write almost everything sequentially, which is an advantage on certain storage media (e.g., (SMR) hard disks).*

Total: 9.0pt

Aufgabe P1: C Grundlagen

Assignment P1: C Basics

a) Was gibt der unten stehende Code bei der Ausführung der Funktion `print_test()` aus? Nehmen Sie an, dass keine andere Funktion in `t` schreibt. **1 pt**

What does the code below print when running the function `print_test()`? Assume that no other function is writing to `t`.

```
struct test {
    int a;
    unsigned b;
    unsigned char c;
};

static struct test t;

void print_test(void) {
    t.c = 65;
    printf("%d/%x/%u\n", t.a, t.b, t.c);
}
```

Solution:

0/0/65

***Note:** Indicating the newline is optional.*

b) Geben Sie für alle Felder des unten stehenden `struct mystruct` die Größe des Feldes und die Größe des Paddings *nach* dem Feld in Bytes an. Schreiben Sie „0“, falls kein Padding eingefügt wird. Gehen Sie von einem 64-Bit-System aus. **1.5 pt**

For each field of the `struct mystruct` below, give the field's size and the size of the padding after *the field in bytes. Write "0" if the compiler does not insert any padding. Assume a 64-bit system.*

Code	Field size [Bytes]	Padding size [Bytes]
`struct mystruct {`	—	—
`  int *a;`	8	0
`  uint16_t b;`	2	6
`};`	—	—

(0.5 P) for two field sizes, **(0.5 P)** per padding size

Warum fügt der Compiler Padding nach den Feldern ein? **1 pt**

Why does the compiler adds padding after struct fields?

Solution:

CPUs can access aligned **(0.5 P)** *memory locations more efficiently due to cache organization in fixed-length cache lines* **(0.5 P)** *or might even disallow unaligned access.*

c) Nehmen Sie an, dass Sie eine Bibliothek entwickeln. Sie haben die beiden unten stehenden Headerdateien `types.h` und `funcs.h` geschrieben. Ein Kunde versucht, Ihre Bibliothek zu verwenden (`main.c`), aber der Code kompiliert nicht. Warum? Gehen Sie davon aus, dass die Includes korrekt aufgelöst werden. **1 pt**

Imagine you are developing a library. You have written the two header files `types.h` and `funcs.h` below. A customer is trying to use your library (`main.c`), but the code does not compile. Why? Assume that the includes resolve correctly.

types.h

```
typedef struct {
    int x, y;
} point;
```

funcs.h

```
#include "types.h"

int distance(point a,
             point b);
```

main.c

```
#include "types.h"
#include "funcs.h"

int main() { /* ... */ }
```

Solution:

The file `types.h` ends up two times in `main.c`, once via the direct include and a second time via `funcs.h`. **(0.5 P)** *The compiler will complain about duplicate definitions of `point`.* **(0.5 P)**

Was müssen Sie zur Datei `types.h` hinzufügen, um das Problem zu lösen? **1.5 pt**

What do you need to add to `types.h` to resolve the issue?

Solution:

```
#ifndef TYPES_H
#define TYPES_H

typedef struct _point {
    int x, y;
} point;

#endif
```

(0.5 P) *per line*

d) In dieser Teilaufgabe sollen Sie ein wachsendes Array (`ga`) implementieren. Es besteht aus einer Länge (`len`), einer Kapazität (`cap`) und einem Pointer zu den separat allozierten Daten (`data`), welche `int`-Werte beinhalten sollen.

In this task, you will implement a growing array (`ga`). It consists of a length (`len`), a capacity (`cap`), and a pointer to separately-allocated data (`data`) which shall contain `int` values.

```
typedef struct _ga {
    size_t len, cap; /* number of elements */
    int *data;
} ga;
```

Warum ist es bei einer solchen Datenstruktur sinnvoll, Länge und Kapazität unabhängig voneinander verändern zu können? **1 pt**

Why does it make sense with such a data structure to be able to change length and capacity independently of each other?

Solution:

With just the length field, every push *operation would need to reallocate the array. With the extra capacity information, we can allocate memory for multiple* push *operations and can thus amortize the cost of reallocation.*

Implementieren Sie die Funktion `ga_init()`, die ein leeres wachsendes Array mit der gegebenen Kapazität alloziert und initialisiert. **3 pt**

- Geben Sie `NULL` zurück, falls ein Fehler auftritt.
- Stellen Sie sicher, dass kein Speicherleck auftritt.
- Der Datenbereich muss nicht initialisiert werden.

Implement the function `ga_init()` that allocates and initializes an empty growing array with the given capacity.

- *Return `NULL` if an error occurs.*
- *Make sure you do not leak any memory.*
- *You do not need to initialize the data buffer.*

Solution:

```
ga *ga_init(size_t cap) {
    ga *a = malloc(sizeof(ga));
    if (a == NULL) return a;
    a->len = 0;
    a->cap = cap;
    a->data = malloc(sizeof(int) * cap);
    if (a->data == NULL) {
        free(a);
        return NULL;
    }
    return a;
}
```

first malloc **(0.5 P)**, *initialize length/cap* **(0.5 P)**, *second malloc* **(0.5 P)**, *error handling* **(1 P)**, *return* **(0.5 P)**

Note: *You do not need to* initialize *the data buffer (e.g., by overwriting it with `memset()`), but you do need to* allocate *memory for it and set the `a->data` pointer accordingly.*

Implementieren Sie die Funktion `ga_push()`, die ein Element an das wachsende Array anhängt. **5 pt**

- Verdoppeln Sie die Kapazität des Arrays wenn nötig.
- Stellen Sie sicher, dass kein Speicherleck auftritt.
- Geben Sie einen Pointer auf das neu angefügte Element zurück, oder `NULL`, wenn ein Fehler auftritt. Verändern Sie `*a` in dem Fall nicht.

Implement the function `ga_push()` that appends an element to the growing array.

- *Double the capacity of the array if necessary.*
- *Make sure you do not leak any memory.*
- *Return a pointer to the inserted item, or `NULL` if an error occured. Leave `*a` unchanged in that case.*

Solution:

```
int *ga_push(ga *a, int d) {
    if (a->cap == a->len) {
        size_t new_cap = a->cap * 2;
        int *new_data = realloc(a->data, sizeof(int) * new_cap);
        if (new_data == NULL) {
            // allocation failed
            return NULL;
        }
        a->cap = new_cap;
        a->data = new_data;
    }

    int *ret = a->data + a->len++;
    *ret = d;
    return ret;
}
```

checking capacity **(0.5 P)**, *calling `realloc()`* **(1 P)**, *error check* **(0.5 P)**, *setting cap and data* **(1 P)**, *calculating pointer to new item* **(0.5 P)**, *setting item* **(0.5 P)**, *incrementing `len`* **(0.5 P)**, *return* **(0.5 P)**

***Alternative** without `realloc()`:*

```
    /* ... */
    if (a->cap == a->len) {
        size_t new_cap = a->cap * 2;
        int *new_data = malloc(sizeof(int) * new_cap);
        if (new_data == NULL) {
            return NULL;
        }
        for (size_t i = 0; i < a->cap; i++) {
            new_data[i] = a->data[i];
        }
        free(a->data);
        a->cap = new_cap;
        a->data = new_data;
    }
    /* ... */
```

Total: 15.0pt

Aufgabe P2: Ausgabeumleitung

Assignment P2: Output Redirection

Sie sollen ein Programm `log_output` schreiben, das ein anderes Programm startet und dessen Ausgabe sowohl auf die Standardausgabe als auch in eine Logdatei umleitet. Wird das Programm zum Beispiel als `log_output ls -l` aufgerufen, so startet es `ls -l` und gibt die Ausgabe sowohl in einer Datei "log.txt" im aktuellen Arbeitsverzeichnis sowie auf der Standardausgabe aus.

- Sie müssen keine C-Header inkludieren.
- Sie müssen keine Fehlerbehandlung implementieren.
- Geben Sie jegliche im Code angeforderten Ressourcen sofern möglich wieder frei. Dies schließt die Freigabe mit `fork()` erstellter Prozesse ein.

You shall write a program `log_output` which starts another program and redirects its output to the standard output as well as into a log file. If, for example, the program is executed via `log_output ls -l`, it starts `ls -l` and writes its output into a file "log.txt" in the current working directory as well as to the standard output.

- *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 whenever possible. This includes freeing resources in processes created via `fork()`.*

a) Weshalb benötigt man in der Praxis meistens nach dem `exec()`-Systemaufruf noch einen Aufruf von `exit()`? **1 pt**

In practice, why do most calls to the `exec()` system call have to be followed by a call to `exit()`?

Solution:

The call is required for error handling: exec() *is usually called after* fork(), *and if the program cannot be executed, the original program would continue running twice, which would cause unwanted side effects or at least increase CPU usage* **(1.0 P)**.

Vervollständigen Sie die Funktion `launch_child()`, die ein Programm startet und einen Dateideskriptor einer Pipe zurückgibt, aus der die Ausgabe des Programms gelesen werden kann. **6 pt**

- Das Programm sowie die Parameter werden in `program` übergeben – wie bei den Parametern für `main()` ist `program[0]` das zu startende Programm.
- `program` ist der Einfachheit halber ein nullterminiertes Array – bei N Parametern (inkl. Programm) ist `program[N]` immer `NULL`.
- Leiten Sie sowohl Standardausgabe (Dateideskriptornummer `STDOUT_FILENO`) als auch Fehlerausgabe (Dateideskriptornummer `STDERR_FILENO`) um.
- Sie können `dup2()` verwenden, um einen Dateideskriptor durch eine Kopie eines anderen zu ersetzen und dadurch Zugriffe auf den Deskriptor umzuleiten.

Complete the function `launch_child()` which starts a program and returns a file descriptor of a pipe from which the output of the program can be read.

- *The program as well as the parameters are passed in* `program` *– as in the case of the parameters for* `main()`*,* `program[0]` *is the program to start.*
- *For reasons of simplicity,* `program` *is a null-terminated array – for N parameters (including the program),* `program[N]` *is always* `NULL`*.*
- *Redirect standard output (file descriptor number* `STDOUT_FILENO`*) as well as error output (file descriptor number* `STDERR_FILENO`*).*
- *You can use* `dup2()` *to replace a file descriptor with a copy of another to redirect accesses to the file descriptor.*

Solution:

```
int launch_child(char **program) {
    int pid;

    /* create a pipe */
    int pipe_fd[2];
    pipe(pipe_fd);

    /* launch the child process */
    pid = fork();
    if (pid == 0) {
        dup2(pipe_fd[1], STDOUT_FILENO);
        dup2(pipe_fd[1], STDERR_FILENO);
        close(pipe_fd[0]);
        close(pipe_fd[1]);
        execvp(program[0], program);
        exit(-1); /* error handling, not necessary */
    }

    close(pipe_fd[1]);
    return pipe_fd[0];
}
```

Create a pipe **(1.0 P)***, fork* **(0.5 P)***, overwrite stdout and stderr* **(1.5 P)** *in the child process* **(0.5 P)***, close unnecessary FDs in the child process* **(0.5 P)** *and the parent process* **(0.5 P)***, execute the program* **(1.0 P)***, return the read end of the pipe* **(0.5 P)***.*

The function must not call `wait()`*. As the pipe buffer is finite, concurrency is required to send arbitrary amounts of data through the pipe.*

b) Vervollständigen Sie die Funktion `redirect_output()`, die sämtliche Daten aus dem übergebenen Dateideskriptor liest und sowohl in eine Datei "log.txt" im aktuellen Arbeitsverzeichnis als auch auf die Standardausgabe (`STDOUT_FILENO`) schreibt. **5 pt**

- Die Datei soll geleert werden, wenn sie bereits existiert. Falls sie noch nicht existiert, soll sie angelegt werden. In diesem Fall soll sie nur für den aktuellen Benutzer les- und schreibbar sein, andere Benutzer sollen keinen Zugriff auf die Datei haben.

Complete the function `redirect_output()` *which reads all data from the specified file descriptor and writes it into a file "log.txt" in the current working directory as well as to the standard output (*`STDOUT_FILENO`*).*

- *The file shall be emptied if it already exists. If it does not exist yet, it shall be created. In this case, only the current user shall be able to read and write the file, other users shall not be able to access the file.*

Solution:

```
void redirect_output(int pipe_fd) {
    char buffer[512];
    int file;
    ssize_t bytes;

    file = open("log.txt", O_CREAT | O_WRONLY | O_TRUNC, 0600);

    while ((bytes = read(pipe_fd, buffer, 512)) != 0) {
        write(file, buffer, bytes);
        write(STDOUT_FILENO, buffer, bytes);
    }

    close(file);
    close(pipe_fd);
}
```

Open the file **(1.0 P)** *with the correct access rights for a new file* **(0.5 P)**, *read into a buffer* **(1.0 P)** *and write the data into the file and stdout* **(1.5 P)**, *repeat until EOF* **(0.5 P)**, *close the file descriptors* **(0.5 P)**.

c) Vervollständigen Sie die Hauptfunktion `main()` des Programms. **3 pt**

- Die Funktion soll das in `argv[1]` spezifizierte Programm mit den Parametern in `argv[2]` bis `argv[argc-1]` starten und die Ausgabe mittels `redirect_output()` umleiten.
- Wenn das gestartete Programm beendet wurde, soll die Funktion den Rückgabewert des Programms ermitteln und selbst zurückgeben.
- Der C-Standard schreibt vor, dass `argv[argc]` immer `NULL` ist – auch dieses Array ist also bereits nullterminiert.

Complete the main function of the program.

- *The function shall start the program specified in* `argv[1]` *with the parameters in* `argv[2]` *to* `argv[argc-1]` *and shall redirect its output using* `redirect_output()`.
- *When the program exits, the function shall determine the return value of the program and shall return it.*
- *The C standard stipulates that* `argv[argc]` *is always* `NULL` *– this array is therefore already null-terminated.*

Solution:

```
int main(int argc, char **argv) {
    int pipe_fd = launch_child(&argv[1]);
    redirect_output(pipe_fd);

    int status;
    wait(&status);

    return WEXITSTATUS(status);
}
```

Call `launch_child()` *with the correct arguments* **(1.0 P)**, *call* `redirect_output()` **(0.5 P)**, *wait for the child* **(0.5 P)**, *return the exit status* **(1.0 P)**. **Total: 15.0pt**

Aufgabe P3: Dynamic Loader

Assignment P3: Dynamic Loader

Um ein Programm zu starten, lädt der Dynamic Loader zuvor nötige Sektionen des Programms und abhängiger Bibliotheken in den Adressraum. Da die Adressen importierter Funktionen (z. B. `print()`) erst nach dem Laden feststehen, verfügt jede Binärdatei über eine Global Offset Table (GOT), welche beim Laden mit den Adressen importierter Symbole initialisiert wird.

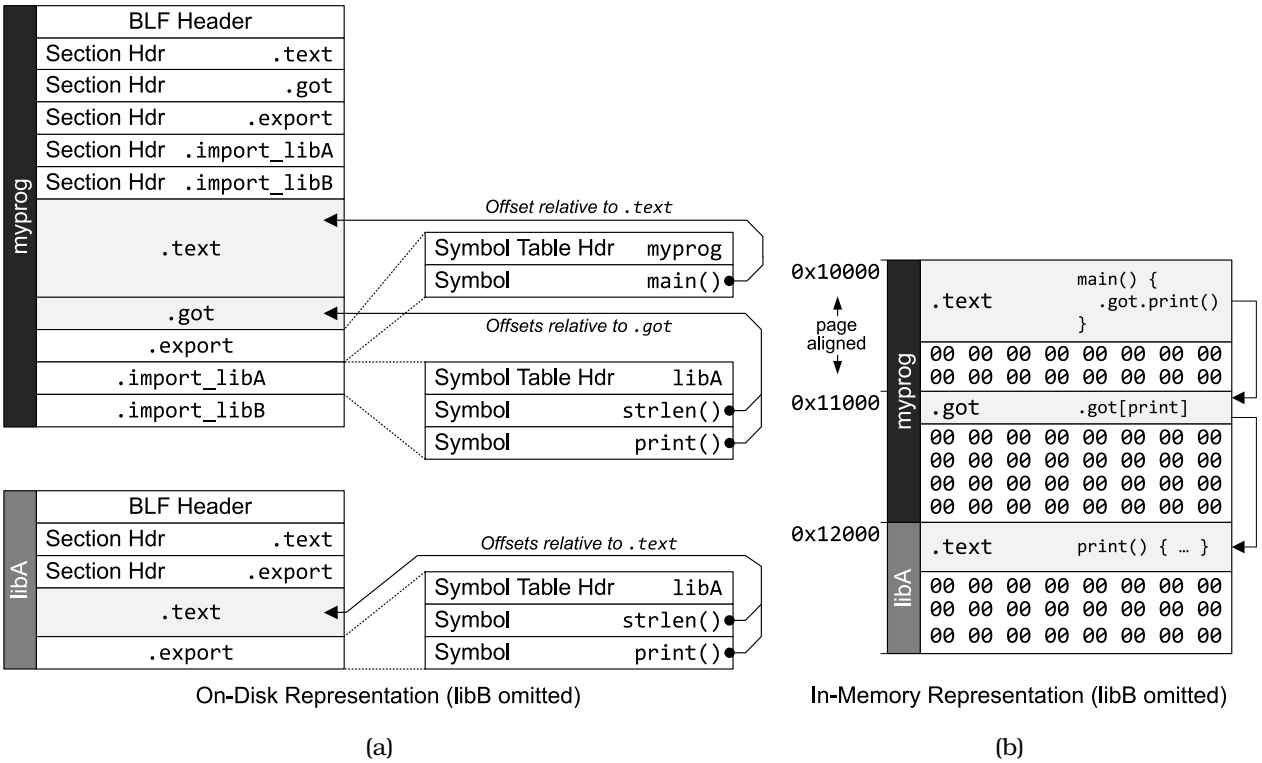
Implementieren Sie einen Dynamic Loader für das Binary Linking Format (BLF):

- Jede Sektion verfügt über einen Header (`Sec`) zu Beginn der BLF-Datei, welcher die Position und Länge der Sektionsdaten angibt.
- Symbole werden in Tabellen (`SymTab`) aufgelistet, welche in den `.export` und `.import` Sektionen gespeichert sind. Für jede importierte Bibliothek existiert eine eigene `.import` Sektion.

To start a program, the dynamic loader first loads necessary sections of the program and dependent libraries into the address space. Since the addresses of imported functions (e.g., `print()`) are not known until after loading, each binary has a Global Offset Table (GOT), which is initialized with the addresses of imported symbols during loading.

Implement a dynamic loader for the Binary Linking Format (BLF):

- *Each section has a header (`Sec`) at the beginning of the BLF file, which specifies the position and length of the section's data.*
- *Symbols are listed in tables (`SymTab`), which are stored in the `.export` and `.import` sections. For each imported library there is a separate `.import` section.*



```c
typedef struct {       // BLF header ----
    char magic[4];     // Magic
    uint8_t nsec;      // Number of sections
} Blf;

const char BLF_MAGIC[] = {'\x7f', 'B', 'L', 'F'};

typedef struct {       // Section header ----
    uint8_t type;      // Section type (-> SEC_*)
    uint32_t foff;     // Offset of section data in file (bytes)
    uint32_t size;     // Size of section data in file (bytes)
} Sec;

#define SEC_TEXT   0  // .text section
#define SEC_GOT    1  // .got section (contains dummy values on disk)
#define SEC_EXPORT 2  // Export symbol table (-> SymTab)
#define SEC_IMPORT 3  // Import symbol table (-> SymTab)

typedef struct {       // Symbol table ----
    char name[16];     // Own name (export) or imported BLF file (import)
                       // - both null-terminated
    uint8_t nsym;      // Number of symbols
} SymTab;

typedef struct {       // Symbol ----
    char name[16];     // Null-terminated symbol name
    uint32_t soff;     // Offset in .text (export) or .got (import)
} Sym;
```

a) Vervollständigen Sie die Funktion `check_magic_blf()`, welche verifiziert, dass das Magic-Feld des übergebenen BLF-Handles mit `BLF_MAGIC` übereinstimmt. **1 pt**

- Geben Sie 0 zurück, wenn keine Übereinstimmung vorliegt, ansonsten einen Wert ungleich 0.

Complete the function `check_magic_blf()` which verifies that the magic field matches `BLF_MAGIC` for the passed BLF handle.

- *Return 0 if there is no match, a value not equal 0 otherwise.*

Solution:

```c
int check_magic_blf(Blf *blf) {
    return !memcmp(blf->magic, BLF_MAGIC, 4);
}
```

performing compare **(0.5 P)**, *returning right value* **(0.5 P)**

Gehen Sie nachfolgend von validen BLF-Dateien und keinen Fehlern aus.
Assume valid BLF files and no errors in the following.

b) Vervollständigen Sie die Funktion `loadsecblf()`, welche die gegebene Sektion `s` des BLF-Programms verarbeitet. Die `.text` und `.got` Sektionen sollen gemäß Abbildung (b) in den Adressraum geladen werden. Für die Symboltabellen soll stattdessen die jeweils passende Funktion aufgerufen werden. **6 pt**

- Nutzen Sie `mmapblf()`, um einen Teil der BLF-Datei in den Adressraum zu laden. `mmapblf()` rundet die Länge auf das nächste Vielfache der Seitengröße (4 KiB) auf und füllt zusätzlichen Raum mit 0-Bytes.
- Laden Sie Sektionen beginnend an `base`. Aktualisieren Sie `base` nach dem Aufruf von `mmapblf()`. Achten Sie darauf, dass `base` stets an Seitengrenzen ausgerichtet ist.
- Speichern Sie die Basisadressen der Sektionen in `sec_base` wie vorgegeben.
- Die Reihenfolge der Sektionen ist durch ihren Typ (`SEC_*`) vorgegeben.
- Sie müssen keine Abhängigkeiten laden.

Complete the function `loadsecblf()` which processes the given section `s` of the BLF program. The `.text` and `.got` sections shall be loaded into the address space according to Figure (b). For the symbol tables, the appropriate function shall be called instead.

- *Use `mmapblf()` to load a part of the BLF file into the address space. `mmapblf()` rounds the length up to the next multiple of the page size (4 KiB) and fills additional space with 0-bytes.*
- *Start loading the sections at `base`. Update `base` after the call to `mmapblf()`. Keep `base` page-aligned.*
- *Store the base addresses of the sections in `sec_base` as specified.*
- *The order of the sections is given by their type (`SEC_*`).*
- *You do not need to load dependencies.*

```
void mmapblf(void *addr, size_t length, Blf *blf, off_t offset);

void *base = (void*)0x10000; // mmap virtual base address

#define ADD_ADDR(a, b) ((void*)((uintptr_t)(a) + (uintptr_t)(b)))
#define NEXT_PAGE(a) (((uintptr_t)(a) + 0x1000 - 1) & ~0xFFFul)

void export_symbols(Blf *blf, off_t symtab, void *text_base);
void import_symbols(Blf *blf, off_t symtab, void *got_base);
```

Solution:

```
void loadsecblf(Blf *blf, Sec *s, void *sec_base[2]) {
    if (s->type <= SEC_GOT) {
        mmapblf(base, s->size, blf, s->foff);
        sec_base[s->type] = base;
        base = ADD_ADDR(base, NEXT_PAGE(s->size));
    } else if (s->type == SEC_EXPORT) {
        export_symbols(blf, s->foff, sec_base[SEC_TEXT]);
    } else
        import_symbols(blf, s->foff, sec_base[SEC_GOT]);
}
```

differentiating by section type **(1.5 P)**, *calling `mmapblf`* **(1.0 P)**, *setting `sec_base`* **(0.5 P)**, *updating `base`* **(0.5 P)** *and making it page aligned* **(0.5 P)**, *calling `export_symbols`* **(1.0 P)**, *calling `import_symbols`* **(1.0 P)**

c) Vervollständigen Sie die Funktion `export_symbols()`, welche für jedes Symbol der Exporttabelle an Offset `symtab` mittels eines Aufrufs von `register_symbol()` die absolute Adresse registriert. **4 pt**

- Nutzen Sie `preadblf()`, um den Symboltabellen-Header zu lesen.
- Nutzen Sie `read_symbol()`, um das Symbol mit dem Index `idx` (beginnt bei 0) aus der Tabelle zu lesen.
- Berechnen Sie die absolute Adresse des Symbols im Adressraum, indem Sie die Basisadresse des Textsegments addieren.

Complete the function `export_symbols()` which registers the absolute address of each symbol in the export table at offset `symtab` via a call to `register_symbol()`.

- *Use `preadblf()` to read the symbol table header.*
- *Use `read_symbol()` to read the symbol at index `idx` (begins at 0) of the table.*
- *Calculate the absolute address of the symbol in the address space by adding the base address of the text segment.*

```
#define ADD_ADDR(a, b) ((void*)((uintptr_t)(a) + (uintptr_t)(b)))

void preadblf(Blf *blf, void *buf, size_t count, off_t offset);
void read_symbol(Blf *blf, off_t symtab, uint8_t idx, Sym *sym);
void register_symbol(Blf *blf, const char *name, void *addr);
```

Solution:

```
void export_symbols(Blf *blf, off_t symtab, void *text_base) {
    SymTab t;
    Sym s;

    preadblf(blf, &t, sizeof(SymTab), symtab);
    for (uint8_t i = 0; i < t.nsym; ++i) {
        read_symbol(blf, symtab, i, &s);

        void *sym_addr = ADD_ADDR(text_base, s.soff);
        register_symbol(blf, s.name, sym_addr);
    }
}
```

loading symbol table **(1.0 P)**, *iterating over symbols* **(1.5 P)**, *adding base address* **(0.5 P)**, *calling `register_symbol`* **(1.0 P)**

d) Vervollständigen Sie die Funktion `import_symbols()`, welche für jedes Symbol der Importtabelle an Offset `symtab` mittels `SET_GOT()` die absolute Symboladresse in der angegebenen GOT ablegt. **4 pt**

- Nutzen Sie `preadblf()`, um den Symboltabellen-Header zu lesen.
- Nutzen Sie `loadblf()`, um die referenzierte BLF-Bibliothek zu laden. Bereits geladene Bibliotheken werden nicht mehrfach geladen.
- Nutzen Sie `read_symbol()`, um das Symbol mit dem Index `idx` (beginnt bei 0) aus der Tabelle zu lesen.
- `lookup_symbol()` liefert die absolute Adresse eines zuvor registrierten Symbols.

Complete the function `import_symbols()` which saves the absolute address of each symbol in the import table at offset `symtab` via `SET_GOT()` in the specified GOT.

- *Use `preadblf()` to read the symbol table header.*
- *Use `loadblf()` to load the referenced BLF library. Libraries that have already been loaded will not be loaded multiple times.*
- *Use `read_symbol()` to read the symbol at index `idx` (begins at 0) of the table.*
- *`lookup_symbol()` returns the absolute address of a registered symbol.*

```
#define SET_GOT(got_base, soff, addr) \
    (*((void**)ADD_ADDR(got_base, soff)) = (addr))

Blf* loadblf(const char *path);
void preadblf(Blf *blf, void *buf, size_t count, off_t offset);

void read_symbol(Blf *blf, off_t symtab, uint8_t idx, Sym *sym);
void* lookup_symbol(Blf *blf, const char *name);
```

Solution:

```
void import_symbols(Blf *blf, off_t symtab, void *got_base) {
    SymTab t;
    Sym s;

    preadblf(blf, &t, sizeof(SymTab), symtab);

    Blf *lib = loadblf(t.name);
    for (uint8_t i = 0; i < t.nsym; ++i) {
        read_symbol(blf, symtab, i, &s);

        void *sym_addr = lookup_symbol(lib, s.name);
        SET_GOT(got_base, s.soff, sym_addr);
    }
}
```

loading symbol table **(0.5 P)**, *calling* `loadblf` **(1.0 P)**, *iterating over symbols* **(0.5 P)** *calling* `lookup_symbol` **(1.0 P)** *using* `SET_GOT` **(1.0 P)** **Total: 15.0pt**