Leibniz Universität Hannover

Klausur Vertiefung der Betriebssysteme

	erreichbare Punkte	erhaltene Punkte
Aufgabe 1	16	
Aufgabe 2	8	
Aufgabe 3	9	
Aufgabe 4	11	
Aufgabe 5	9	
Aufgabe 6	10	
Aufgabe 7	12	
Aufgabe 8	8	
Aufgabe 9	6	
Summe	89	

(Name) (Vorname) (Matrikel-Nr.)

(Studiengang) (Semester)

Durch meine Unterschrift bestätige ich
- den Empfang der vollständigen Klausur (16 Seiten inklusive Deckblatt),
- den Empfang der Manualseiten (accept, bind, clone, close, exit, fork, futex, listen, mmap, read, socket und write),
- die Kenntnisnahme der Hinweise auf Seite 2.

Hannover, 22.08.2023

(Unterschrift)

Hinweise

Bitte lesen Sie die folgenden Informationen aufmerksam und unterschreiben Sie die Erklärung auf der ersten Seite.

- Die Bearbeitungszeit beträgt 90 Minuten.
- Es sind **keine** eigenen Hilfsmittel zugelassen.
- Die Lösung einer Aufgabe soll auf das Aufgabenblatt in den dafür vorgesehenen Raum geschrieben werden. Beachten Sie bitte, dass der freigelassene Platz großzügig bemessen ist und nicht unbedingt der erwarteten Antwortlänge entspricht. Sollte der Platz nicht ausreichen, können Sie die Rückseiten der Aufgabenblätter mitverwenden. Kennzeichnen Sie dabei die Zugehörigkeit Ihrer Lösung zu einer Aufgabe. Bei Bedarf können zusätzliche Lösungsblätter (weiß) ausgeteilt werden. Vermerken Sie vor deren Verwendung unbedingt Ihren Namen und Ihre Matrikelnummer darauf!
- Die Lösungen müssen dokumentenecht in blau oder schwarz geschrieben werden. Als falsch Erkanntes muss deutlich durchgestrichen werden. Tintenkiller und andere Korrekturstifte dürfen nicht verwendet werden. Keinen Bleistift verwenden!
- Fragen zu den Prüfungsaufgaben können grundsätzlich **nicht** beantwortet werden.
- Tragen Sie Ihren Namen und Vornamen, Ihre Matrikelnummer, Studiengang und Fachsemesterzahl auf dem Deckblatt der Klausur ein.
- Tragen Sie auf jedem Blatt Ihren Namen ein.
- Um Unruhe und die Störung Ihrer Mitstudierenden zu vermeiden, ist die vorzeitige Abgabe der Klausur ausgeschlossen. Bleiben Sie an Ihrem Platz sitzen, bis am Ende alle Klausurunterlagen eingesammelt sind und die Aufsicht das Zeichen zum Gehen gibt.

Aufgabe 1: (16 Punkte)

In dieser Aufgabe sind jeweils m Aussagen angegeben. Davon sind n ($0 \leq n \leq m$) Aussagen richtig. Kreuzen Sie jeweils an, ob die entsprechende Aussage richtig oder falsch ist.
Jede korrekte Antwort gibt 1 Punkt, jede falsche Antwort 1 Punkt Abzug. Nicht beantwortete Aussagen gehen neutral in die Bewertung ein. Eine Teilaufgabe wird minimal mit 0 Punkten gewertet, falsche Antworten wirken sich nicht auf andere Teilaufgaben aus.

Wollen Sie eine falsch angekreuzte Antwort korrigieren, streichen Sie bitte das Kreuz mit drei waagrechten Strichen durch (☒).

Lesen Sie die Frage genau, bevor Sie antworten.

a) Isolation und Sicherheit

4 Punkte

Richtig Falsch

- ☐ ☐ Mit eBPF lassen sich beliebige Anwendungen im Kernel-Space ausführen, wodurch diese Vollzugriff auf das System bekommen und schneller laufen.
- ☐ ☐ Durch Maßschneiderung des Betriebssystems können potentielle Angriffsvektoren reduziert werden.
- ☐ ☐ Ein Privilegierter Prozess kann seine user-id beliebig verändern.
- ☐ ☐ `chroot` verändert die Sicht eines Prozesses auf das Dateisystem.

b) Dateien und Rechte

4 Punkte

```
root@node001:/test# ls -lah
total 36K
drwxr-xr-x  2 root         root   4,0K 21. Aug 14:30 .
drwx------ 11 root         root   4,0K 21. Aug 14:26 ..
-rwxr-sr-x  1 cybercat     floppy  17K 21. Aug 14:26 editor
-rw-rw----  1 pixelpenguin floppy   13 21. Aug 14:28 passphrase
-rwxr-xr-t  1 root         root    17K 21. Aug 14:39 secret
----r--rw-  1 cybercat     floppy  118 21. Aug 14:26 secret.c

root@node001:/test# id blazebear
uid=1010(blazebear) gid=1010(blazebear) groups=1010(blazebear)
```

Richtig Falsch

- ☐ ☐ Das Unix Rechtesystem verbietet dem User `cybercat` explizit das Lesen der Datei `secret.c`.
- ☐ ☐ Das *sticky bit* der `secret`-Datei sorgt dafür, dass der ausführende Prozess Privilegien des Datei-Eigentümers erhält.
- ☐ ☐ Jeder kann durch Ausführen des `editor`-Programms auf die `passphrase`-Datei schreibend zugreifen.
- ☐ ☐ Das Windows (NT) Rechtesystem ermöglicht das explizite Verbieten von Dateizugriffsrechten.

c) Skalierbarkeit

4 Punkte

Richtig	Falsch	
☐	☐	Realtime-Scheduling erhöht den Durchsatz eines Rechnersystems.
☐	☐	Bei einem Benchmark sind andere Arbeitslasten auf dem Rechner vernachlässigbar, da der Benchmark die gesamte Rechenleistung beansprucht und somit die anderen verdrängt.
☐	☐	Durch vollständige Implementierung der Synchronisation im Betriebssystemkern sind SystemV-Semaphore besonders performant.
☐	☐	Aufgrund der Isolationsgarantien eines Betriebssystems beeinflussen unterschiedliche Prozesse ihr Laufzeitverhalten keinesfalls gegenseitig.

d) Systemaufrufe: Semantik und Implementierung

4 Punkte

Richtig	Falsch	
☐	☐	Der `clone()` Systemaufruf bietet deutlich mehr Flexibelität als `fork()`, wodurch unter Aufgabe bestimmter Eigenschaften eine bessere Performance erreicht werden kann.
☐	☐	*Huge Pages* verringern den Verwaltungsaufwand für virtuellen Speicher im Betriebssystem.
☐	☐	Die POSIX-Semantik verbietet eine skalierbare Implementierung des `dup()`-Systemaufrufs.
☐	☐	`writev()` verbessert den Schreibdurchsatz durch die erhöhte Zahl an benötigten Systemaufrufen.

Aufgabe 2: Virtualisierung (8 Punkte)

a) Erklären Sie die Unterschiede zwischen Virtualisierung und Paravirtualisierung und nennen Sie jeweils einen Vorteil.

4 Punkte

b) Wie unterscheiden sich Container (z.B. Docker) von virtuellen Maschinen (z.B. Qemu/KVM)? Nennen Sie jeweils einen Vorteil für beide Varianten.

4 Punkte

Aufgabe 3: (9 Punkte)

Implementieren Sie die Basisoperationen eines Semaphors unter Ausnutzung des Futex-Konzepts und des dazugehörigen `futex()`-Systemaufrufs.

Hinweis: Einfache Lese- und Schreiboperationen auf Integer-Datentypen sind atomar.

```
// Atomare Operationen, erklärt in C-Pseudocode
int faa(atomic_int* value, int increase) {
  atomic {
    int old = *value;
    *value += increase;
    return old;
  }
}

int cas(atomic_int* value, int old, int new) {
  atomic {
    if (*value == old) {
      *value = new;
      return true;
    } else {
      return false;
    }
  }
}

int tas(atomic_int* value) {
  atomic {
    if (*value == 0) {
      value = 1;
      return 0;
    } else {
      return 1;
    }
  }
}
```

```
void futex_sem_up(atomic_int *sem) {
```

U:

```
void futex_sem_down(atomic_int *sem) {
```

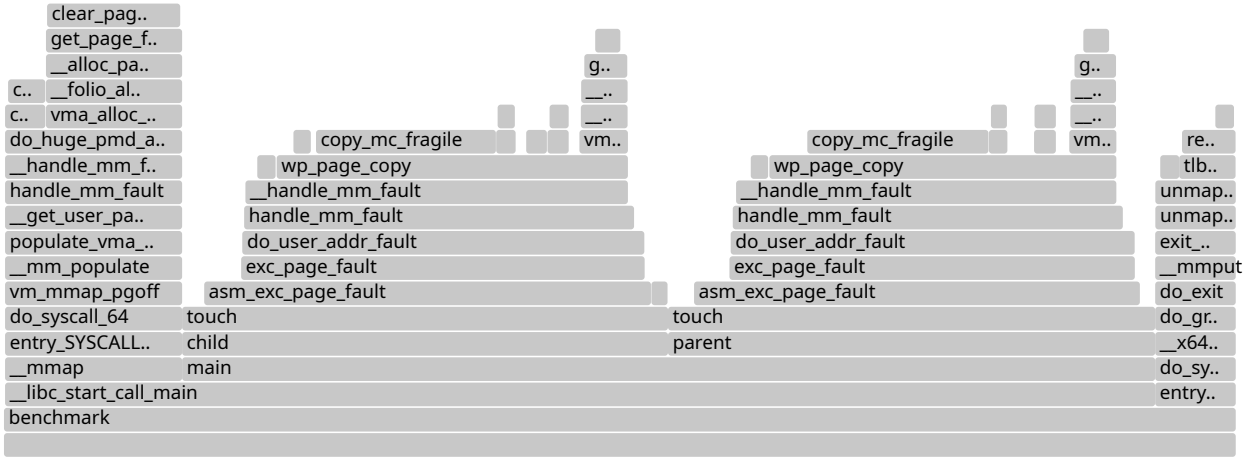
D:

Aufgabe 4: Messen und Interpretieren (11 Punkte)

Der abgebildete Programmcode implementiert zwei verschiedene Varianten, um einen neuen Prozess zu erzeugen und jeweils im Eltern- und Kindprozess eine andere Funktion auszuführen. Für eine Variante wurde ein Flamegraph erstellt.

Hinweis: `MAP_POPULATE` sorgt dafür, dass direkt bei der Erstellung des Mappings der gesamte Bereich mit Speicherseiten gefüllt wird.

```
#define LEN (4UL << 30)
char stack[4096];

// Write a single byte to every page
void touch(char* s, int c, size_t n) {
  for (unsigned long i = 0; i < n; i += 4096) {
    s[i] = c;
  }
}

int child(void* arg) {
  touch(arg, 41, LEN);
  return 0;
}

int parent(void* arg) {
  touch(arg, 42, LEN);
  return 0;
}

int main() {
  void* ptr = mmap(0, LEN,
    PROT_READ | PROT_WRITE,
    MAP_PRIVATE | MAP_ANONYMOUS | MAP_POPULATE,
    -1, 0);

#ifdef FORK
  pid_t pid = fork();
  if (pid == 0) {
    child(ptr);
  } else {
    parent(ptr);
  }
#else
  pid_t pid = clone(child, stack + sizeof(stack) - 1, CLONE_VM, ptr);
  parent(ptr);
#endif

  exit(0);
}
```



a) Beschreiben Sie die 4 Hauptkomponenten des Flamegraphs.

4 Punkte

b) Ordnen Sie dem Flamegraph eine Programmvariante zu und begründen Sie ihre Antwort.

3 Punkte

c) Beschreiben Sie, wie der Flamegraph für die andere Variante aussehen müsste. Begründen Sie ihre Antwort.

3 Punkte

d) Wie müsste `clone()` in dem gezeigten Beispielcode verwendet werden, um die gleiche Funktionalität wie `fork()` zu erreichen?

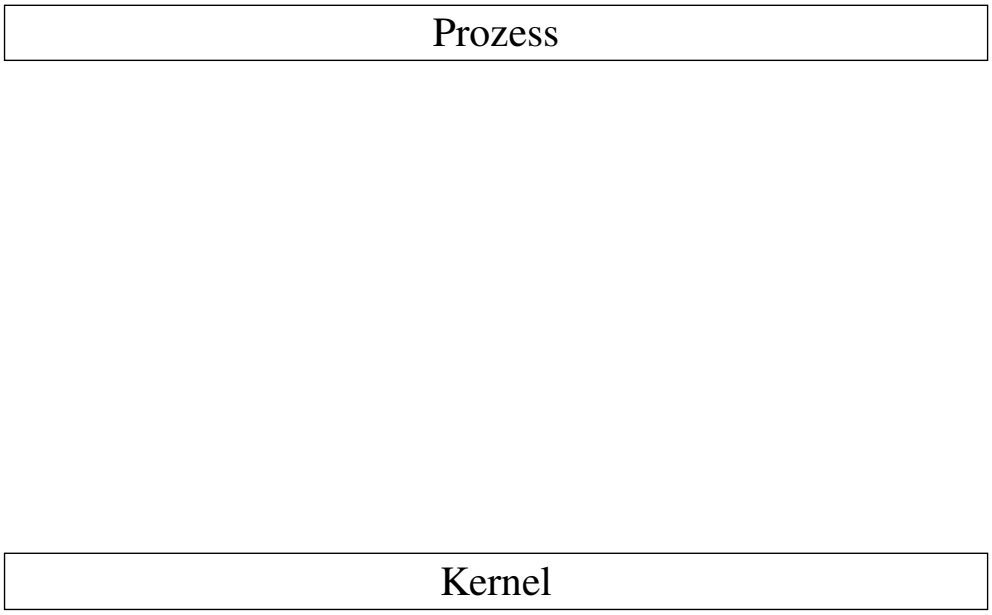
1 Punkt

Aufgabe 5: IO_Uring (9 Punkte)

9 Punkte

a) Erklären Sie das hinter io_uring stehende Konzept.

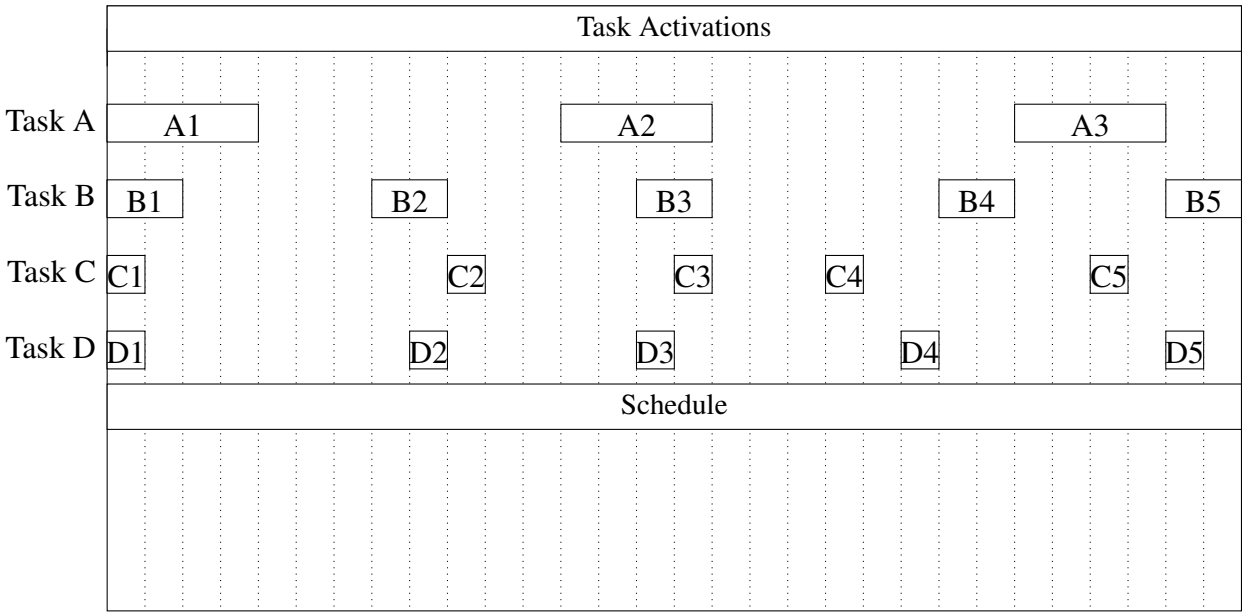
- Vervollständigen Sie dazu die folgende Skizze,
- benennen Sie die entscheidenen Komponenten
- und beschreiben Sie die Funktionsweise.



Aufgabe 6: Echtzeitverarbeitung (10 Punkte)

a) Gegeben seien die folgenden Aktivierungen und Rechenbedarfe der Tasks A, B, C und D. Die Deadline ist jeweils das nächste Auslösen des gleichen Tasks. Zeichnen sie den Ausführungsplan (Schedule) für ein Einkernsystem nach dem FIFO-Verfahren in die untere Hälfte des Schaubildes ein. Markieren und benennen Sie eventuell auftretende Probleme oder den Planungserfolg.
Hinweis: Bei gleicher Ankunftszeit werden die Tasks alphabetisch sortiert abgearbeitet.

4 Punkte



b) Nennen Sie ein alternatives Schedulingverfahren und jeweils einen Nach- und Vorteil gegenüber dem FIFO-Verfahren.

3 Punkte

c) Welchen Einfluss hat ein *Big Kernel Lock* auf die Echtzeitfähigkeit eines Betriebssystems? Was folgt daraus?
Nennen Sie außerdem eine Alternative.

3 Punkte

Aufgabe 7: Interprozesskommunikation (12 Punkte)

a) Das folgende Programm soll einen einfachen Echoserver implementieren, der auf Port 5027 auf eingehende TCP-Verbindungen wartet. Um möglichst viele Nutzer bedienen zu können, wird **nur eine Nachricht** (maximal 4 KiB) empfangen und zurück gesendet bevor die Verbindung wieder getrennt und auf die nächste Verbindung gewartet wird.

8 Punkte

Hinweise:

- Lese- und Schreiboperationen werden immer vollständig durchgeführt.
- Auf Fehlerbehandlung kann an dieser Stelle verzichtet werden.
- Ergänzen Sie das folgende Codegerüst zu einem vollständig übersetzbaren Programm.

```
int main() {
    struct sockaddr_in addr = {
        .sin_family = AF_INET,
        .sin_addr.s_addr = htonl(INADDR_ANY),
        .sin_port = htons(5027)
    };
    ____________________________________________________________;
    ____________________________________________________________;
    ____________________________________________________________;
    while (true) {
        char buffer[4096];
        ____________________________________________________________;
        ____________________________________________________________;
        ____________________________________________________________;
        ____________________________________________________________;
    }
    return 0;
}
```

b) Nennen Sie **zwei** Alternativen für **systemlokale** Interprozesskommunikation und jeweils einen Vorteil gegenüber TCP.

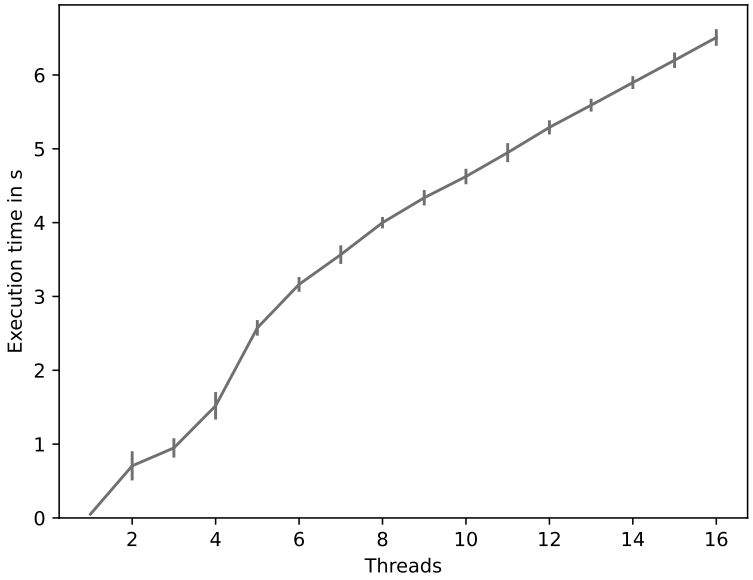
4 Punkte

Aufgabe 8: Skalierbarkeit und Synchronisierung (8 Punkte)

Das dargestellte Programm simuliert einen Produktionsablauf, bei dem unterschiedliche Ressourcen jeweils unabhängig voneinander verbraucht werden. Die Ausführung mit einer unterschiedlichen Anzahl an parallelen Abläufen und 10^6 Einheiten pro Ressource erzeugt das abgebildete Diagramm.

Hinweis: Eine Cacheline umfasst 64 Byte

```
int32_t counters[16] __attribute__((aligned(64)));
sem_t lock;

void* consumer(void* arg) {
  int index = (int)arg;
  bool run = true;
  while (run) {
    sem_wait(&lock);
    int32_t val = --counters[index];
    sem_post(&lock);
    if (val <= 0)
      run = false;
  }
  return NULL;
}

int main(int argc, char *argv[]) {
  if (argc != 3) {
    printf("Usage: %s <threads> <items>\n", argv[0]);
    return -1;
  }

  int thread_count = atoi(argv[1]);
  int items = atoi(argv[2]);

  for (int i = 0; i < thread_count; i++)
    counters[i] = items;

  pthread_t threads[16];
  for (int i = 0; i < thread_count; i++)
    pthread_create(&threads[i], NULL, consumer, (void*)i);

  for (int i = 0; i < thread_count; i++)
    pthread_join(threads[i], NULL);

  return 0;
}
```



a) Was ist die Ursache für die schlechte Parallelisierbarkeit? Begrüden Sie ihre Antwort.

2 Punkte

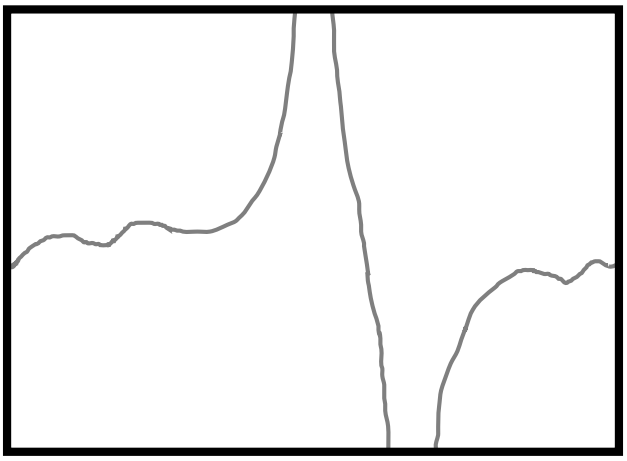
b) Wie kann die Parallelisierbarkeit verbessert werden?
Nennen Sie **drei Verbesserungsmaßnahmen** und begründen Sie ihre Antworten.

6 Punkte

Aufgabe 9: Statistik und Visualisierung (6 Punkte)

a) Das dargestellte Diagramm zeigt die Latenz einer Serveranwendung. Welche Aspekte der Darstellung sollten verbessert werden? Nennen Sie drei Punkte.

3 Punkte



b) Die Messung von Softwarelaufzeiten unterliegt einer Vielzahl von Einflussfaktoren. Nennen Sie **drei** Einflussfaktoren und Methoden, um diese zu reduzieren.

3 Punkte