

System-Programmierung (syspr) 03. Juni, 2025

thomas. amberg@fhnw.ch

Assessment II

vorname:	Punkte: / 90,	Note:
Name:	Frei lassen für Korrekt	ur.
Klasse: 4ibb1		
Hilfsmittel:		
- Ein A4-Blatt handgeschriebene Zusammenfassung.		
- Lösen Sie die Aufgaben jeweils direkt auf den Prüfungs	sblättern.	
- Zusatzblätter, falls nötig, mit Ihrem Namen und Frage	-Nr. auf jedem Blatt.	
Nicht erlaubt:		
- Unterlagen (Slides, Bücher,).		
- Computer (Laptop, Smartphone,).		
- Kommunikation (mit Personen, KI,).		
Bewertung:		
- Multiple Response: \Box <i>Ja</i> oder \Box <i>Nein</i> ankreuzen, +1/	-1 Punkt pro richtige/fals	sche Antwort,
beide nicht ankreuzen ergibt +0 Punkte; Total pro Fra	ge gibt es nie weniger als	0 Punkte.
- Offene Fragen: Bewertet wird Korrektheit, Vollständig	keit und Kürze der Antw	ort.
- Programme: Bewertet wird die Idee/Skizze und Umset	tzung des Programms.	
Fragen zur Prüfung:		
- Während der Prüfung werden vom Dozent keine Frage	en zur Prüfung beantwor	tet.

- Ist etwas unklar, machen Sie eine Annahme und notieren Sie diese auf der Prüfung.



Threads und Synchronisation

1) Welche der folgenden Aussagen über Threads sind korrekt? Punkte:			Punkte: _ / 4	
Zutreffendes ankre	ruzen			
□ Ja □ Nein	Zwei Threads in einen	n Prozess führen verschiedene Progr	amme aus.	
□ Ja □ Nein	Der Main-Thread kan	Der Main-Thread kann andere Threads mit pthread_join() beenden.		
□ Ja □ Nein	Ein beliebiger Thread kann sich und alle anderen mit <i>exit()</i> beenden.			
\square Ja \square Nein	Jeder neue Thread sieht eine eigene Kopie aller globalen Variablen.		ariablen.	
Welche der folge	enden Aussagen über Cri	tical Sections sind korrekt?	Punkte: _ / 4	
Zutreffendes ankre	ruzen			
\square Ja \square Nein	Eine Critical Section muss immer atomar ausgeführt werden.			
\square Ja \square Nein	Eine Critical Section garantiert den gegenseitigen Ausschluss.			
□ Ja □ Nein	Ein einzelnes C Statement kann eine Critical Section sein.			
□ Ja □ Nein	Kritisch ist die Ausführungsreihenfolge der Instruktionen.			
3) Was sind drei we	esentliche Unterschiede z	zwischen Mutex und Semaphoren?	Punkte: _ / 6	
Unterschiede hier e	eintragen, jeweils beide S	Seiten in einem kurzen Satz beschre	iben:	
Mutex		Semaphore		

IPC mit Pipes

4) Schreiben Sie ein Programm, das zwei Child-Prozesse via Pipe verbindet, ein per Command Line übergebenes Wort von Child 1 zu Child 2 sendet und dort auf *stdout* ausgibt. P.kte: _ / 18 *Verwenden Sie die folgenden Calls (soweit sinnvoll)*, ohne #includes und Fehlerbehandlung:

```
void exit(int status); // cause process termination; does not return
pid_t fork(void); // create a child process, returns 0 in child process

int pipe(int pipe_fd[2]); // create a pipe, from pipe_fd[1] to pipe_fd[0]

int close(int fd); // close a file descriptor
ssize_t read(int fd, void *buf, size_t n); // attempts to read up to n
bytes from file descriptor fd into buf; returns number of bytes read ≤ n
ssize_t write(int fd, const void *buf, size_t n); // writes up to n bytes
from buf to the file referred to by fd; returns nr. of bytes written ≤ n
size_t strlen(const char *s); // calculate the length of a string
```

Sockets

5) Schreiben Sie ein Programm, welches UNIX Domain Byte-Streams unter der Adresse /log empfängt, und die gelesenen, max. 1024 Bytes auf STDOUT_FILENO schreibt. Punkte: _ / 16 Verwenden Sie die folgenden Calls (soweit sinnvoll), ohne #includes und Fehlerbehandlung:

```
int accept(int socket, struct sockaddr *addr, socklen_t *addrlen); //
accept a connection on a socket; returns the accepted client socket
int bind(int socket, struct sockaddr *addr, socklen_t addrlen); // bind
int listen(int socket, int backlog); // listen for connections on socket
int socket(int domain, int type, int protocol); // create an endpoint for
communication; domain = AF_INET, AF_UNIX; type = SOCK_STREAM, SOCK_DGRAM;
protocol = 0; returns a file descriptor for the new socket or -1 on error

struct sockaddr_un { // UNIX domain socket address structure
    sa_family_t sun_family; // AF_UNIX
    char sun_path[108]; // Pathname
};

int close(int fd); // close a file descriptor
ssize_t read(int fd, void *buf, size_t count); // read from a file descr.
ssize_t write(int fd, const void *buf, size_t count); // write to a file
char *strcpy(char *dest, char *src); // copy a string src to buffer dest
```

6) Welche Aussagen zu Internet Datagram Sockets treffen im Allgemeinen zu? Punkte: _/4				
Zutreffendes ankr	euzen:			
□ Ja □ Nein	Datagram Sockets sind	Datagram Sockets sind verbindungslos, es braucht daher kein <i>connect()</i> .		
□ Ja □ Nein	Zu jeder Message gibt e	es zwei IP-Adressen, Empfänger und Absender.		
□ Ja □ Nein	Der Zugriff auf Interne	Der Zugriff auf Internet Datagram Sockets ist via Filesystem geregelt.		
□ Ja □ Nein	Die gängigere Bezeichn	Die gängigere Bezeichnung für Internet Datagram Sockets ist TCP.		
POSIX IPC				
7) Was sind drei w	vesentliche Unterschiede v	von POSIX Message Queue zu Pipes? Punkte: _ / 6		
Unterschiede hier	eintragen, jeweils beide	Seiten in einem kurzen Satz beschreiben:		
POSIX Message Queue		Pipe		
1				
1				
· 				

Fortsetzung auf der nächsten Seite ...

8) Schreiben Sie *ein* Programm, das 9 Personen beim Ausverkauf simuliert, die gleichzeitig versuchen, eins von 5 Shirts zu erhalten. Nutzen Sie *fork()* und eine Named Semaphore /*sem*. Personen, die eins erhalten, geben ihre *PID* und "yes" aus, sonst *PID* und "no". Punkte: _ / 14 *Verwenden Sie die folgenden Calls, ohne #includes, mit Fehlerbehandlung, soweit wie nötig:*

```
int printf(const char *format, ...); // format string %s, char %c, int %d

void exit(int status); // cause process termination; does not return
pid_t getpid(void); // returns the process ID (PID) of the calling proc.
pid_t fork(void); // create a child process, returns 0 in child process

sem_t *sem_open(char *name, int oflag, mode_t mode, unsigned int val); //
create, initialize and open a named semaphore; returns a pointer to the
semaphore; use oflag = O_CREAT; mode = S_IRUSR|S_IWUSR and set its value
int sem_post(sem_t *s); // increment a semaphore
int sem_wait(sem_t *s); // decrement a semaphore, blocks if value <= 0
int sem_trywait(sem_t *sem); // returns 0 on success, -1 if value <= 0</pre>
```

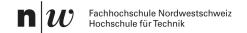
Zeitmessung

9) Schreiben Sie ein Programm *yesterday*, welches das korrekte Kalenderdatum von gestern, oo:oo:oo, in Lokalzeit, im default Format ausgibt, wie hier im Beispiel gezeigt. Punkte: / 10

```
$ date
Sat May 31 19:22:40 CEST 2025
$ ./yesterday
Fri May 30 00:00:00 2025
```

Verwenden Sie die folgenden Calls (soweit sinnvoll), ohne #includes und Fehlerbehandlung:

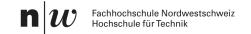
```
int printf(const char *format, ...); // format string %s, char %c, int %d
char *ctime(const time_t *t); // t to string, local time, default format
struct tm *localtime(const time_t *t); // get broken-down local time
time_t mktime(struct tm *tm); // convert broken-down local time to time_t
; ignores tm_wday, tm_yday; values outside valid interval are normalised
time_t time(time_t *t); // returns time in seconds since 01.01.1970, UTC
struct tm {
  int tm_sec; // Seconds (0-60)
                                     int tm_year; // Year - 1900
                                     int tm_wday; // Day of the week
  int tm_min; // Minutes (0-59)
  int tm_hour; // Hours (0-23)
                                                  // (0-6, Sunday = 0)
  int tm_mday; // Day of the
                                     int tm_yday; // Day in the year
               // month (1-31)
                                                   // (0-365, 1 Jan = 0)
               // Month (0-11)
                                     int tm_isdst; // Daylight saving time
  int tm_mon;
                                   };
  . . .
```



Zusatzblatt auf der nächsten Seite ...

10) Welche der folge	enden Aussagen zu Zeitmessung treffen im Allgemeinen zu? Punkte: $_/4$
Zutreffendes ankre	uzen:
□ Ja □ Nein	Die Linux Epoche ist die Zeit seit dem Aufstarten.
□ Ja □ Nein	User CPU Zeit ist immer grösser als System CPU Zeit.
□ Ja □ Nein	CPU Zeit wird relativ gemessen, an mehr als einem Punkt.
□ Ja □ Nein	Reale Zeit ist die Summe von User und System CPU Zeit.
Terminals	
11) Welche der folge	enden Aussagen zu Terminals treffen im Allgemeinen zu? Punkte: _ / 4
Zutreffendes ankre	uzen:
□ Ja □ Nein	Echo führt dazu, dass der User-Prozess jedes Zeichen zweimal sieht.
□ Ja □ Nein	Im Canonical Mode können User Tippfehler in der Shell korrigieren.
□ Ja □ Nein	Text-Editoren wie <i>nano</i> lesen Terminal Input jeweils Zeile für Zeile.
□ Ja □ Nein	Eine Änderungen der Terminal Fenstergrösse löst ein Signal aus.
I	
l	
İ	

Seite 8 / 9



Zusatzblatt zu Aufgabe Nr	von (Name)	