

System-Programmierung (syspr) 08. Juni 2023

thomas.amberg@fhnw.ch

## **Assessment II**

Vorname:	Punkte: / 90,	Note:
Name:	Frei lassen für Korrek	tur.
Klasse: 4ibb2		
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/fal	lsche Antwort,
beide nicht ankreuzen ergibt +0 Punkte; Total pro Fra	ge gibt es nie weniger al	s 0 Punkte.
- Offene Fragen: Bewertet wird Korrektheit, Vollständig	keit und Kürze der Antv	vort.
- Programme: Bewertet wird die Idee/Skizze und Umse	tzung des Programms.	
Fragen zur Prüfung:		
- Während der Prüfung werden vom Dozent keine Frage	en zur Prüfung beantwoi	rtet.

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

## Threads und Synchronisation

1) Schreiben Sie ein Programm  $sleep\_sort$ , das Zahlen sortiert, indem es für jede übergebene Zahl n in einem eigenen Thread sleep(n) aufruft, und dann die Zahl ausgibt. Punkte: \_\_\_ / 14

```
$ ./sleep_sort 4 2 1 5
1 2 4 5
```

Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
int atoi(const char *nptr); // convert a string to an integer
int printf(const char *format, ...); // format string %s, char %c, int %d

int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
   void *(*start) (void *), void *arg); // starts a thread; attr = NULL
int pthread_join(pthread_t thread, void **retval); // retval = NULL
int sleep(int seconds); // calling thread sleeps for a number of seconds
```

Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Frage-Nr.:

2) Welche der folg	enden Aussagen über Threads sind korrekt?	Punkte: _ / 4
Zutreffendes ankr	reuzen	
□ Ja   □ Nein	Zwei Threads in einem Prozess führen verschiedene F	Programme aus.
□ Ja   □ Nein	Der Main-Thread kann andere Threads mit <i>pthread</i> _	join() beenden.
□ Ja   □ Nein	Ein beliebiger Thread kann sich und alle anderen mit	exit() beenden.
□ Ja   □ Nein	Jeder neue Thread sieht eine eigene Kopie aller globa	len Variablen.
IPC mit Pipe	es	
3) Schreiben Sie ei	in Programm, das zwei Child-Prozesse via Pipe verbindet	c, ein per Command
Line übergebenes	Wort von Child 1 zu Child 2 sendet und dort auf <i>stdout</i> a	usgibt. P.kte: _ / 18
Hier ein Auszug a	us der Doku, #includes und Fehlerbehandlung können S	lie weglassen:
I .	status); // cause process termination; does nod); // create a child process, returns 0 in c	
1	<pre>pipe_fd[2]); // create a pipe, from pipe_fd[1] fd); // close a file descriptor</pre>	to pipe_fd[0]
bytes from fil ssize_t write(	Int fd, void *buf, size_t n); // attempts to release terms number of being into form the size_t n); // writes ne file referred to by fd; returns nr. of byte	ytes read ≤ n up to n bytes
size_t <b>strlen</b> (	const char *s); // calculate the length of a	string
Idee (kurz) und So	ource Code hier und auf Folgeseite, oder Blatt mit Name	n & Frage-Nr.:

(3) Fortsetzung:	
Sockets	
4) Welche der folge	enden Aussagen über Internet Domain Sockets sind korrekt? Punkte: _ / 4
Zutreffendes ankre	euzen:
□ Ja   □ Nein	Der bind() Aufruf nimmt beides, Internet und Unix Domain Adressen.
□ Ja   □ Nein	Internet Domain Sockets erlauben Datentransfer zwischen Unix Hosts.
$\square$ Ja   $\square$ Nein	File Permissions bestimmen, wer auf Internet Sockets zugreifen kann.
$\square$ Ja   $\square$ Nein	Internet Domain Datagram Sockets übertragen Messages zuverlässig.
5) Wenn Sie http:/	//fhnw.ch/ im Browser öffnen, wer ruft <i>read()</i> auf, und wozu? Punkte: _ / 4
ssize_t read(i	nt fd, void *buf, size_t count); // read bytes from socket
Freitext Antwort u	and kurze Begründung:

## **POSIX IPC**

6) Schreiben Sie ein Programm seat, das mit einer POSIX Message Queue eine Warteschlange für Restaurants umsetzt, die erlaubt, Wartende mit seat wait name eat | drink aufzunehmen und später mit seat next zu sehen, wer den nächsten Sitzplatz bekommt. Personen, die essen (eat), sollen dabei solche überholen, die nur Drinks wollen (drink), wie gezeigt. Punkte: \_ / 16

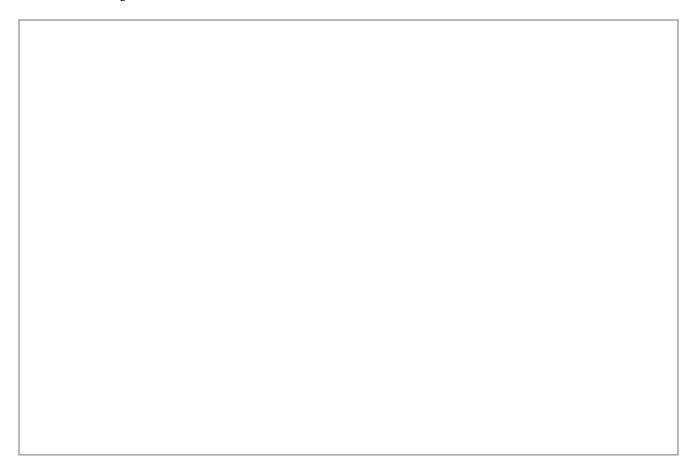
```
$ ./seat wait barney drink
added barney to queue
$ ./seat wait homer eat
added homer to queue
$ ./seat next
next is homer
```

Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
mqd_t mq_open(char *name, int flags, mode_t mode, struct mq_attr *attr);
// open a message queue or create it; returns a message queue descriptor;
flags include O_RDONLY, O_WRONLY, O_RDWR and O_CREAT; mode incl. S_IRUSR
and S_IWUSR; struct mq_attr { long mq_maxmsg, long mq_msgsize, ... };
int mq_send(mqd_t mqd, char *msg, size_t len, unsigned int prio);
// send a message to a message queue; returns 0 on success
ssize_t mq_receive(mqd_t mqd, char *msg, size_t len, unsigned int *prio);
// receive a message from a message queue; returns # of bytes in message
int mq_close(mqd_t mqd); // close a message queue descriptor
int mq_unlink(const char *name); // remove a message queue
int printf(const char *format, ...); // format string %s, char %c, int %d
int strcmp(const char *s1, const char *s2); // compare two strings;
returns 0 if the strings s1 and s2 are equal
size_t strlen(const char *s); // calculate the length of a string
```

Idee (kurz) und Source Code hier und auf Folgeseite, oder Blatt mit Namen & Frage-Nr.:

## (6) Fortsetzung:



7) Schreiben Sie ein Programm, das 9 Personen beim Ausverkauf simuliert, die gleichzeitig versuchen, eins von 5 T-Shirts zu ergattern. Nutzen Sie dazu Threads und Semaphore, damit exakt 5 Personen ein Shirt bekommen. Diese geben "yes" aus, alle anderen "no". P.kte: \_ / 14 Hier die Doku, #includes und nicht-essentielle Fehlerbehandlung können Sie weglassen:

```
int printf(const char *format, ...); // format string %s, char %c, int %d
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
    void *(*start) (void *), void *arg); // starts a thread; attr = NULL
int pthread_detach(pthread_t thread); // detach a thread
void pthread_exit(void *retval); // terminate calling thread, no return

int sem_init(sem_t *sem, int pshared, unsigned int value); // initialize
an unnamed semaphore, pshared = 0
int sem_wait(sem_t *s); // decrement a semaphore, blocking if <= 0
int sem_trywait(sem_t *sem); // returns 0 on success, -1 if sem is locked
int sem_post(sem_t *s); // increment a semaphore</pre>
```

Idee (kurz) und Source Code auf Folgeseite, oder Blatt mit Namen & Fragen-Nr.

(7) Fortsetzung:		
<b>7</b>		
Zeitmessun	g	
7) Welche der folg	genden Aussagen zu Zeitmessung treffen im Allgemeinen zu?	Punkte: _ / 4
Zutreffendes ank	reuzen:	
□ Ja   □ Nein	Die Linux Epoche ist die Zeit seit dem Aufstarten.	
□ Ja   □ Nein	User CPU Zeit ist immer kleiner als System CPU Zeit.	
□ Ja   □ Nein	Reale Zeit ist die Summe von User und System CPU Zeit.	
□ Ja   □ Nein	CPU Zeit wird relativ gemessen, an mehr als einem Punkt.	

8) Schreiben Sie ein Programm, das eine Sekunde lang (CPU Zeit) Child-Prozesse erzeugt und dann ausgibt, wie viele Child-Prozesse in dieser Zeit erzeugt werden konnten. Punkte: \_ / 12 Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
clock_t clock(void); // determine CPU time; resolution is CLOCKS_PER_SEC

void exit(int status); // cause process termination; does not return

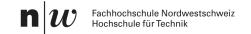
pid_t fork(void); // create a child process, returns 0 in child process

pid_t wait(int *wstatus); // wait for child process to terminate; returns

the process ID of the terminated child or -1 if no child left to wait for

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

dee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Frage-Nr.:				



Zusatzblatt zu Aufgabe Nr	von (Name)	