

System-Programmierung (syspr) 06. Juni 2023

thomas. amberg@fhnw.ch

Assessment II

Vorname:	Punkte: / 90,	Note:
Name:	Frei lassen für Korrekt	tur.
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	n-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	sche Antwort,
beide nicht ankreuzen ergibt +0 Punkte; Total pro Fra	ge gibt es nie weniger als	s 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) Schreiben Sie ein Programm add, das n als Command Line Argumente übergebene Zahlen in n Threads zu int konvertiert, aufaddiert, und in main() das Resultat ausgibt. Punkte: $_/$ 16

```
$ ./add 3 4 2
9
```

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 pthread_mutex_lock(pthread_mutex_t *mutex); // lock a mutex
int pthread_mutex_unlock(pthread_mutex_t *mutex); // unlock a mutex
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER; // initialize a mutex
```

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



IPC mit Pipes

2) Was sind drei w	esentliche Unterschiede zwi	schen Pipes und FIFOs?	Punkte: _ / 6
Unterschiede hier	eintragen, jeweils beide Seit	ten in einem kurzen Satz beschro	eiben:
Pipe		FIF0	
Sockets			
3) Welche der folg	enden Aussagen über Unix I	Domain Sockets sind korrekt?	Punkte: _ / 4
Zutreffendes ankr	euzen:		
□ Ja □ Nein	Unix Domain Sockets erl	auben Kommunikation zwischer	n Unix Hosts.
□ Ja □ Nein	Der bind() Aufruf nimmt beides, Internet und Unix Domain Adressen.		
□ Ja □ Nein	File Permissions bestimn	File Permissions bestimmen, wer Zugriff auf Unix Domain Sockets hat.	
□ Ja □ Nein	Unix Domain Datagram S	Sockets übertragen Datenpakete	zuverlässig.

4) Schreiben Sie ein Programm, das mit *socketpair()* zwei verbundene UNIX Domain Sockets erstellt und diese nutzt, um ein (per Command Line übergebenes) ASCII-Zeichen vom Parentzum Child-Prozess und wieder zurück zu senden, wo es dann ausgegeben wird. Punkte: _ / 14 *Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:*

void exit(int status); // cause process termination; does not return pid_t fork(void); // create a child process, returns 0 in child process int printf(const char *format, ...); // format string %s, char %c, int %d int socketpair(int domain, int type, int prot, int fds[2]); // create a pair of connected sockets; domain = AF_UNIX; type = SOCK_STREAM; prot = 0; sockets fds[0] and fds[1] are indistinguishable; returns 0 on success 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

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

POSIX IPC

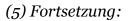
5) Schreiben Sie ein Programm *checkin*, das mit einer *POSIX Message Queue* eine Flughafen Check-in Warteschlange umsetzt, die es erlaubt, Reisende mit *checkin scan name eco|biz* zu erfassen und später mit *checkin next* herauszufinden, wer als nächstes dran kommt. Business Reisende sollen dabei solche mit Economy Ticket überholen, wie hier gezeigt. Punkte: _ / 16

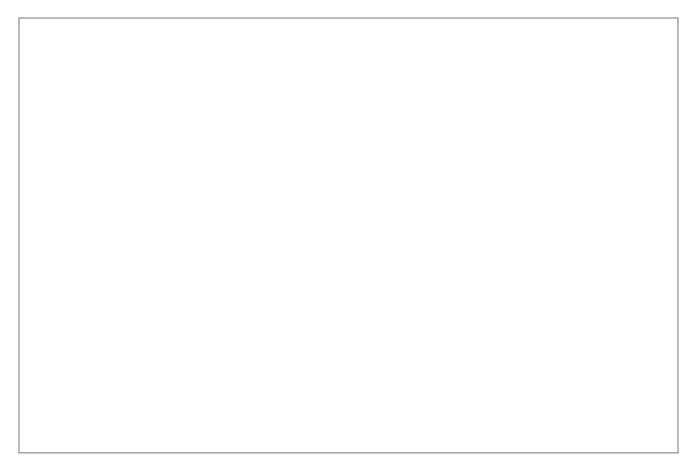
```
$ ./checkin scan bart eco
added bart (eco) to queue
$ ./checkin scan lisa biz
added lisa (biz) to queue
$ ./checkin next
next is lisa
```

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
```

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





6) Schreiben Sie ein Programm, das 7 Personen simuliert, die gleichzeitig versuchen, in ein Taxi mit 3 freien Plätzen einzusteigen. Nutzen Sie Threads und garantieren Sie, dass am Ende exakt 3 Fahrgäste im Taxi sind. Diese geben "got in!" aus, alle anderen "too late". 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.

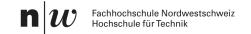
(6) Fortsetzung:		
Zeitmessun	g	
7) Welche der folg	genden Aussagen zu Zeitmessung treffen im Allgemeinen zu?	Punkte: _ / 4
Zutreffendes ank	reuzen:	
\square Ja \square Nein	Die Linux Epoche ist die Zeit seit dem Aufstarten.	
□ Ja □ Nein	User CPU Zeit ist immer kleiner als System CPU Zeit.	
□ Ja □ Nein	Reale Zeit ist die Summe von User und System CPU Zeit.	
\square Ja \square 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:

<pre>clock_t clock(void); // determine CPU time; resolution is CLOCKS_PER_SEC</pre>
<pre>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</pre>
int printf (const char *format,); // format string %s, char %c, int %d
Idee (kurz) und Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Frage-Nr.:

Terminals

9) Welche der fol	genden Aussagen zu Terminals treffen im Allgemeinen zu?	Punkte: _ / 4
Zutreffendes ank	reuzen:	
\square Ja \square Nein	Im Canonical Mode können User Tippfehler in der Shell k	orrigieren.
\square Ja \square Nein	Echo führt dazu, dass der User-Prozess jedes Zeichen zwe	imal sieht.
\square Ja \square Nein	Text-Editoren wie nano lesen Terminal Input jeweils Zeile	e für Zeile.
□ Ja □ Nein	Eine Änderungen der Terminal Fenstergrösse löst ein Sigi	nal aus.



Zusatzblatt zu Aufgabe Nr	von (Name)	