

# Assessment II

Vorname: \_\_\_\_\_

Punkte: \_\_\_\_ / 90, Note: \_\_\_\_

Name: \_\_\_\_\_

*Frei lassen für Korrektur.*

Klasse: 4ibb1

Hilfsmittel:

- Ein A4-Blatt handgeschriebene Zusammenfassung.
- Lösen Sie die Aufgaben jeweils direkt auf den Prüfungsblättern.
- 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: ☐ *Ja* oder ☐ *Nein* ankreuzen, +1/-1 Punkt pro richtige/falsche Antwort, beide nicht ankreuzen ergibt +0 Punkte; Total pro Frage gibt es nie weniger als 0 Punkte.
- Offene Fragen: Bewertet wird Korrektheit, Vollständigkeit und Kürze der Antwort.
- Programme: Bewertet wird die Idee/Skizze und Umsetzung des Programms.

Fragen zur Prüfung:

- Während der Prüfung werden vom Dozent keine Fragen zur Prüfung beantwortet.
- Ist etwas unklar, machen Sie eine Annahme und notieren Sie diese auf der Prüfung.

## Threads und Synchronisation

1) Schreiben Sie ein Programm *sum*, das *n* per Command Line übergebene Zahlen parallel, in *n* Threads, zu *int* konvertiert und aufaddiert, dann in *main()* die Summe ausgibt. P.kte: \_ / 16

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

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

```
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 C Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Frage-Nr.:

## IPC mit Pipes

2) Schreiben Sie ein Programm *sd*, welches die maximale *Tiefe des Stacks* bestimmt, indem es eine rekursive Funktion aufruft, in einem Child Prozess, welche in jeder Iteration die aktuelle Stack-Tiefe per Pipe an den Parent schickt, der dann den letzten Wert ausgibt. Punkte: \_ / 18

*Hinweis: Um einen int n zu schreiben, genügt write(fd, &n, sizeof(int)), dasselbe bei read().*

```
$ ./sd
524119 recursive calls before stack overflow
```

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

```
int printf(const char *format, ...); // format string %s, char %c, int %d
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]
ssize_t read(int fd, void *buf, size_t count); // read from a file descr,
returns 0 (EOF) if reading a pipe which has been closed on the other end
ssize_t write(int fd, const void *buf, size_t count); // write to a file
int close(int fd); // close a file descriptor, returns 0 on success
```

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

## Sockets

3) Wenn Sie <http://fhnw.ch/> im Browser öffnen, wer ruft *write()* auf, und wozu? P.kte: \_ / 4

```
ssize_t write(int fd, const void *buf, size_t count); // write to socket
```

*Antwort in ein/zwei Sätzen hier, oder auf Zusatzblatt mit Ihrem Namen und Frage-Nr.:*

4) Welche der folgenden Aussagen über Unix Domain Sockets sind korrekt? Punkte: \_ / 4

*Zutreffendes ankreuzen:*

- |   |   |
|---|---|
| <input type="checkbox"/> Ja   <input type="checkbox"/> Nein | Unix Domain Sockets erlauben Kommunikation zwischen Unix Hosts.           |
| <input type="checkbox"/> Ja   <input type="checkbox"/> Nein | Der <i>bind()</i> Aufruf nimmt beides, Internet und Unix Domain Adressen. |
| <input type="checkbox"/> Ja   <input type="checkbox"/> Nein | File Permissions bestimmen, wer Zugriff auf Unix Domain Sockets hat.      |
| <input type="checkbox"/> Ja   <input type="checkbox"/> Nein | Unix Domain Datagram Sockets übertragen Datenpakete zuverlässig.          |

|

|

|

|

*Fortsetzung auf der nächsten Seite.*

## POSIX IPC

5) Schreiben Sie ein Programm *mq\_carousel*, welches alle Messages aus einer per Pfadname gegebenen POSIX Message Queue herausliest, auf die Konsole ausgibt, und dann wieder in die Queue schreibt. Vereinfachung: Alle Messages haben dieselbe Priorität x. Punkte: \_ / 12

```
$ ./mq_carousel /my_mq
hello
hola
hi
```

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

```
int mq_getattr(mqd_t mqd, struct mq_attr *attr); // read attributes
struct mq_attr { ... long mq_maxmsg; long mq_msgsize; long mq_curmsgs; };
// max # of messages; max message size; current number of messages on mq
mqd_t mq_open(const char *name, int flag); // open a message queue, flag
// is exactly one of O_RDONLY, O_WRONLY, O_RDWR for send, receive or both
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_send(mqd_t mqd, char *msg, size_t len, unsigned int prio);
// send a message to a message queue; returns 0 on success

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

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

6) Schreiben Sie ein Programm, das immer 3 Child-Prozesse hat, welche zufällig nach *rand()* Sekunden terminieren. Ein *Named Semaphor* soll die Anzahl konstant halten. Punkte: \_ / 12  
Verwenden Sie die folgenden Calls (soweit sinnvoll), ohne *#includes* und Fehlerbehandlung:

```
int rand(); // returns a random number [0, RAND_MAX]
```

```
void exit(int status); // cause normal process termination  
pid_t fork(void); // create a child process; returns 0 in child process;  
// the termination signal of the child is always SIGCHLD  
pid_t wait(int *wstatus); // wait for process to change state
```

```
sem_t *sem_open(char *name, int oflag, mode_t mode, unsigned int value);  
// initialize and open a named semaphore; O_CREAT, ...; S_IRUSR, S_IWUSR, ...  
int sem_post(sem_t *s); // increment a semaphore  
int sem_wait(sem_t *s); // decrement a semaphore, blocking if <= 0
```

```
int sleep(int seconds); // calling thread sleeps for a number of seconds
```

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

## Zeitmessung

7) Schreiben Sie ein Programm *tomorrow*, welches "morgen", 00:00:00 als Datum (gemäss Kalender) ausgibt, im Standardformat des Betriebssystems, wie hier im Beispiel. P.kte: \_ / 12

```
$ date
Tue Jun  4 13:37:11 2024
$ ./tomorrow
Wed Jun  5 00:00:00 2024
```

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

```
int printf(const char *fmt, ...); // fmt int %d, double %lf, string %s

char *ctime(const time_t *t); // convert t to ASCII, default date 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); // get local time in seconds since Epoch

struct tm {
    int tm_sec, tm_min, tm_hour; // (0-60), (0-59), (0-23)
    int tm_mday; // Day of the month (1-31)
    int tm_mon; // Month (0-11)
    int tm_year; // Year - 1900
    int tm_wday; // Day of the week (0-6, Sunday = 0)
    int tm_yday; // Day in the year (0-365, 1 Jan = 0)
};
```

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





## Terminals

9) Welche der folgenden Aussagen zu Terminals treffen im Allgemeinen zu? Punkte: \_ / 4

*Zutreffendes ankreuzen:*

- ☐ 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 nano lesen Terminal Input jeweils Zeile für Zeile.
- ☐ Ja | ☐ Nein      Eine Änderungen der Terminal Fenstergrösse löst ein Signal aus.

*Zusatzblatt zu Aufgabe Nr. \_\_\_\_ von (Name) \_\_\_\_\_*

*Zusatzblatt zu Aufgabe Nr. \_\_\_\_ von (Name) \_\_\_\_\_*