

# Assessment II

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

Punkte: \_\_\_\_ / 76, Note: \_\_\_\_

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

*Frei lassen für Korrektur.*

Klasse: ☒ Klasse 4ibb1 ☐ Klasse 4ibb2

Hilfsmittel:

- Die vom Dozent ausgeteilte C Referenzkarte.
- Lösen Sie die Aufgaben direkt auf den Prüfungsblättern.
- Zusatzblätter, falls nötig, mit Ihrem Namen und Fragen-Nr. auf jedem Blatt.

Nicht erlaubt:

- Unterlagen (Slides, Bücher, ...).
- Computer (Laptop, Smartphone, ...).
- Kommunikation mit anderen Personen.

Bewertung:

- Multiple Choice: Eine ☒ Antwort pro Frage ankreuzen, 4 Punkte pro richtige Antwort.
- Offene Fragen: Bewertet wird Korrektheit, Vollständigkeit und Kürze der Antwort.
- Programme: Bewertet wird die Skizze/Idee 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 & Synchronisation

1) Beschreiben Sie die Funktionsweise und Mechanismen dieses Programms. Punkte: \_\_\_ / 6

Was simuliert das Programm, wie funktioniert es, und wie heisst das allgemeine Pattern?

```
01 #define MACHINE_MAX_COFFEES 3
02
03 pthread_mutex_t machine_mutex = PTHREAD_MUTEX_INITIALIZER;
04
05 void *barista_start(void *arg) {
06     while (1) {
07         pthread_mutex_lock(&machine_mutex);
08         if (machine_n_coffees() < MACHINE_MAX_COFFEES) {
09             int c = grind_coffee();
10             machine_brew_coffee(c);
11         }
12         pthread_mutex_unlock(&machine_mutex);
13     }
14 }
15
16 void *waiter_start(void *arg) {
17     while (1) {
18         pthread_mutex_lock(&machine_mutex);
19         if (machine_n_coffees() > 0) {
20             int c = machine_remove_coffee();
21             serve_coffee(c);
22         }
23         pthread_mutex_unlock(&machine_mutex);
24     }
25 }
26
27 ... // implementations of app specific functions, machine_..., etc.
28
29 int main () ... // calls barista_start, waiter_start in a thread each
```

*Freitext Antwort hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:*

2) Schreiben Sie den folgenden Code um, so dass `stack_push()` Thread-safe ist. Punkte: \_ / 6

*Thread-safe* heisst hier "ohne Race Conditions" falls mehrere Threads die Funktion aufrufen.

```
01 typedef struct node {
02     struct node *next;
03     int item;
04 } Node;
05
06 static Node *stack = NULL;
07
08 void stack_push(int item) {
09     Node *n = malloc(sizeof(Node));
10     n->item = item;
11     n->next = stack;
12     stack = n;
13 }
14
15 int stack_pop() { ... } // ignore
```

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

```
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

void *malloc(size_t size); // allocate dynamic memory; thread-safe
```

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

## IPC mit Pipes

3) Schreiben Sie ein Programm, das seine Child Prozesse per Pipe verbindet. Punkte: \_\_\_ / 12

Child Prozess Nr. 2 soll die Nachricht "hi" an Child Prozess Nr. 1 senden, über diese Pipe.

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

```
int close(int fd); // close a file descriptor
pid_t fork(void); // create a child process, returns 0 in child process
int pipe(int pipe_fd[2]); // create pipe, from pipe_fd[1] to pipe_fd[0]
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
pid_t wait(int *status); // wait for child process, status can be NULL
```

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

## Sockets

4) Vergleichen Sie Internet Stream Sockets mit Internet Datagram Sockets. Punkte: \_\_ / 6

| <i>Internet Stream Sockets (TCP)</i> | <i>Internet Datagram Sockets (UDP)</i> |
|--------------------------------------|--|
|                                      |  |
|                                      |  |
|                                      |  |

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

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

*Freitext Antwort hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:*

6) Welche Reihenfolge haben die folgenden Calls bei einem UDP Server?

Punkte: \_ / 4

```
int bind(int sock_fd, const struct sockaddr *addr, socklen_t addrlen);  
ssize_t recvfrom(int fd, void *buf, size_t len, int flags,  
    struct sockaddr *addr, socklen_t *addr_len);  
ssize_t sendto(int fd, const void *buf, size_t len, int flags,  
    const struct sockaddr *dest_addr, socklen_t addrlen);  
int socket(int domain, int type, int protocol);
```

*Eine Antwort ankreuzen:*

- ☐ socket(); bind(); sendto(); recvfrom();
- ☐ bind(); socket(); sendto(); recvfrom();
- ☐ socket(); sendto(); bind(); recvfrom();
- ☐ bind(); sendto(); socket(); recvfrom();
- ☐ socket(); bind(); recvfrom(); sendto();
- ☐ bind(); socket(); recvfrom(); sendto();

## Terminals

7) Welcher Terminal Mode wird für Text-Editoren verwendet, und wieso?

Punkte: \_\_\_\_ / 3

*Freitext Antwort hier, oder auf Zusatzblatt mit Ihrem Namen und Fragen-Nr.:*

## POSIX IPC

8) Schreiben Sie ein Programm, das den Verkauf von 99 Tickets durch 7 Verkäufer simuliert.

Verkäufe (parallel) dauern 1-3s, ein Semaphor soll garantieren, dass es 99 sind. Punkte: \_ / 14

*Hier ein Auszug aus der Doku, #includes und unerwartete Fehler können Sie weglassen:*

```
// all return 0 on success, -1 on error, stored in global variable errno
int sem_destroy(sem_t *s); // destroy an unnamed semaphore
int sem_init(sem_t *s, int pshared, unsigned int value); // pshared = 0
int sem_post(sem_t *s); // increment a semaphore
int sem_wait(sem_t *s); // decrement a semaphore, blocking if <= 0
int sem_trywait(sem_t *s); // decrement a semaphore, EAGAIN if <= 0
```

```
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 n); // n seconds
```

```
int rand(void); // returns [0..RAND_MAX]
```

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

## Zeitmessung

9) Schreiben Sie ein Programm, das misst, wie lange (Echtzeit) ein Signal hat. Punkte: \_\_\_ / 12

Die Messung soll beim Senden von *SIGUSR1* starten und direkt nach dem Eintreffen stoppen.

```
$ ./my_signal_time
Sending signal ...
... handling done.
real: 67999 10^-9s
```

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

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

typedef void (*sighandler_t)(int); // handler signature
sighandler_t signal(int signum, sighandler_t handler); // install handler
int raise(int sig); // send a signal to the caller, returns 0 or non-zero

int clock_gettime(clockid_t c_id, struct timespec *s); // returns 0 or -1
// c_id = CLOCK_REALTIME or CLOCK_MONOTONIC or CLOCK_PROCESS_CPUTIME_ID
struct timespec {
    time_t tv_sec; // seconds
    long tv_nsec; // nanoseconds
};
```

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



10) Schreiben Sie ein Programm, das "gestern", 00:00:00 als Datum ausgibt. Punkte: \_\_\_ / 9

Der Output des Programms soll so aussehen, mit dem aktuellen Datum, im Default-Format:

```
$ ./yesterday
today: Sun Jun  2 16:40:44 2019
yesterday: Sat Jun  1 00:00:00 2019
```

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

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

time_t time(time_t *t); // get local time in seconds since Epoch
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

struct tm {
    int tm_sec; // Seconds (0-60)
    int tm_min; // Minutes (0-59)
    int tm_hour; // Hours (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)
    int tm_isdst; // Daylight saving time
};
```

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

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