

Assessment I

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.

Erste Schritte in C

1) Welche der folgenden Typen sind Teil der Sprache C?

Punkte: _ / 4

Zutreffendes ankreuzen, (C = C99, ohne #includes):

- ☐ Ja | ☐ Nein byte
- ☐ Ja | ☐ Nein size_t
- ☐ Ja | ☐ Nein double
- ☐ Ja | ☐ Nein void *

2) Welche dieser C Ausdrücke sind auf Linux Systemen allgemein wahr?

Punkte: _ / 4

Zutreffendes ankreuzen, (C = C99, mit #includes):

- ☐ Ja | ☐ Nein sizeof(unsigned int) == sizeof(int)
- ☐ Ja | ☐ Nein sizeof(size_t) == sizeof(long)
- ☐ Ja | ☐ Nein sizeof(char) < sizeof(int)
- ☐ Ja | ☐ Nein sizeof(int) == INT_MAX

3) Zu welcher Situation im Speicher führt diese Abfolge von Statements?

Punkte: _ / 4

```
int i = 3;
int *p = &i;
int **q = &p;
// <- Situation?
```

Zutreffende Antwort ankreuzen:

<input type="checkbox"/> Situation A	<input type="checkbox"/> Situation B	<input type="checkbox"/> Situation C	<input type="checkbox"/> Situation D
<p>i: 3</p> <p>p: []</p> <p>q: []</p>	<p>i: 3</p> <p>p: []</p> <p>q: []</p>	<p>i: 3</p> <p>p: []</p> <p>q: []</p>	<p>i: 3</p> <p>p: []</p> <p>q: []</p>

Funktionen in C

4) Schreiben Sie ein Programm *max*, welches das längste der *n* übergebenen Argumente, oder wie ganz unten, das erste von mehreren maximal langen Argumenten, ausgibt. Punkte: _ / 12

```
$ ./max
$ ./max short looong
looong
$ ./max short words use them
short
```

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

```
int printf(const char *format, ...); // format string %s, char %c, int %d
size_t strlen(const char *s); // calculate the length of a string
```

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

|

|

|

Fortsetzung auf der nächsten Seite.

5) Gegeben den folgenden Code, implementieren Sie die Funktion *join()*, die ein neues Buch vorne in die unsortierte Liste einfügt, und diese zurückgibt. Sowie die Funktion *find()*, welche das Buch mit ISBN-Nummer *isbn* in der Liste findet und dieses zurückgibt. Punkte: _ / 12

```
struct book { char title[32]; long isbn; struct book* next; };

struct book *books = NULL;

struct book *join(struct book *list, long isbn, char *title); // TODO
struct book *find(struct book *list, long isbn); // TODO

int main() {
    books = join(books, 9781400075997, "Turing's Cathedral");
    books = join(books, 9781805260981, "The Algorithm");
    books = join(books, 9780593241837, "Unmasking AI");
    struct book *b = find(books, 9781805260981);
    printf("%ld: %s\n", b->isbn, b->title);
}
```

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

```
void *malloc(size_t n); // Allocates n bytes, returns pointer to memory.

char *strcpy(char *dest, char *src); // Copies src to dest, incl. '\0'.
```

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

File In-/Output

6) Schreiben Sie ein Programm *ncopy*, das eine Quelldatei in eine variable Anzahl Zieldateien kopiert. Die Dateinamen übergibt man per Command Line, wie im Beispiel. Punkte: _ / 14

```
$ echo "hello" > my.txt
$ ./ncopy my.txt your.txt their.txt
$ ls *.txt
my.txt  their.txt  your.txt
```

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

```
int open(const char *pathname, int flags, mode_t mode); // Opens the file
specified by pathname. Or creates it if O_CREAT is used. Returns the file
descriptor. Flags include O_APPEND, O_CREAT, O_TRUNC, O_RDONLY, O_WRONLY.
Modes, which are used together with O_CREAT, include S_IRUSR and S_IWUSR.
```

```
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.:

Prozesse und Signale

7) Gegeben den folgenden Code, wie sieht der Prozess-Baum am Schluss aus? Punkte: _ / 4

```
#include ... // ignore

int main() {
    fork(); fork();
    printf("%d\n", getpid());
    // <- Baum?
}
```

Zutreffende Antwort ankreuzen:

<input type="checkbox"/> Baum A	<input type="checkbox"/> Baum B	<input type="checkbox"/> Baum C	<input type="checkbox"/> Baum D
<pre> * \ \ \ * * \ \ 1 3 2 4 </pre>	<pre> * \ * \ \ \ 1 3 2 </pre>	<pre> * \ 1 2 </pre>	<pre> * \ 2 : * \ 1 n </pre>

|

|

|

|

|

|

|

|

|

Fortsetzung auf der nächsten Seite.

8) Schreiben Sie ein Programm *nsig*, das beim *n*-ten Auftreten des Signals *sig* terminiert. Die Parameter *n* und *sig* werden per Command Line übergeben, wie im Beispiel. Punkte: _ / 10

```
$ ./nsig 3 2  
^C^C^C$
```

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

```
int atoi(const char *s); // convert a string to an integer  
  
int pause(void); // sleep until a signal causes invocation of a handler  
  
typedef void (*sighandler_t)(int); e.g. SIGINT = 2, ^C; SIGTSTP = 20, ^Z  
sighandler_t signal(int sig, sighandler_t h); // set h to handle a signal
```

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

|

|

|

Fortsetzung auf der nächsten Seite.

Prozess Lebenszyklus

9) Schreiben Sie ein Programm *run*, welches zwei per Command Line übergebene Programme startet und den Namen des zuerst beendeten Programms wie im Beispiel ausgibt. P.kte: __ / 14

Hinweis: Die wait() Funktion gibt die Prozess-ID des terminierten Child-Prozesses zurück.

```
$ ./run ps ls
... # ignore program output
ls wins.
```

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

```
int execvp(char *file, char *argv[]); // executes the program referred to
by pathname; returns only if an error has occurred; argv can be NULL.
```

```
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
```

```
pid_t wait(int *status); // wait for any child process to terminate;
returns the PID of the terminated child or -1 if no child is left to wait
for; the macro WEXITSTATUS(status) returns the exit status of the child.
```

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

Threads und Synchronisation

10) Gegeben den folgenden Code, der zwei Schalter simuliert, die insgesamt genau n Konzert-Tickets verkaufen, implementieren Sie die `sell()` Funktion. Falls keine Tickets mehr verfügbar sind ($n == 0$), gibt der jeweilige Thread seine ID plus " done." aus, und stoppt. Punkte: / 12

```
#include ... // ignore

int n;
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;

void *sell(void *arg); // TODO

int main(int argc, char *argv[]) {
    n = atoi(argv[1]);
    pthread_t t1, t2;
    pthread_create(&t1, NULL, sell, NULL);
    pthread_create(&t2, NULL, sell, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
```

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

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

```
int pthread_mutex_lock(pthread_mutex_t *mutex); // lock a mutex, returns
0 on success; If the mutex object is already locked by another thread,
the calling thread shall block until the mutex becomes available.
```

```
int pthread_mutex_trylock(pthread_mutex_t *mutex); // lock a mutex, non-
blocking, returns 0 on success, EBUSY if the mutex could not be acquired
because it was already locked.
```

```
int pthread_mutex_unlock(pthread_mutex_t *mutex); // unlock a mutex,
returns 0 on success; If there are threads blocked on the mutex object,
scheduling policy shall determine which thread shall acquire the mutex.
```

```
pthread_t pthread_self(void); // obtain ID of the calling thread
```

Fortsetzung auf der nächsten Seite.

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

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____