

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 Fragen-Nr. auf jedem Blatt.

Nicht erlaubt:

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

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 Aussagen sind korrekt?

Punkte: _ / 4

Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein Variablen des Typs *float* kann man Werte des Typs *byte* zuweisen.
- ☐ Ja | ☐ Nein Ein *int* Wert ist im Speicher des Computers maximal 4 Byte gross.
- ☐ Ja | ☐ Nein Der *sizeof* Operator liefert für den Typ *char* immer das Resultat 1.
- ☐ Ja | ☐ Nein Der Typ *int* benötigt gleich viele Bytes wie der Typ *unsigned int*.

2) Welche Abfolge von Statements führt zu folgender Situation im Speicher?

Punkte: _ / 4



Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein `int p = 2; int q = 0; int *a[] = {&p, &q}; **(&a + 1) = 3;`
- ☐ Ja | ☐ Nein `int q = 2; int *a[] = {0, &q}; int p = q; a[0] = &p; q++;`
- ☐ Ja | ☐ Nein `int p; int q = 2; int *a[2] = {0}; p = q; a[0] = &p; q++;`
- ☐ Ja | ☐ Nein `int p; int q; int *a[] = {&p, &q}; *a[0] = 2; q = p + 1;`

|
|
|
|
|
|
|

(Aufgabe 3 ist auf der nächsten Seite)

Funktionen in C

3) Schreiben Sie ein Programm *shuffle*, welches $n > 0$ übergebene Command Line Argumente in zufälliger Reihenfolge, wie im Beispiel angedeutet, auf die Konsole ausgibt. Punkte: / 12

```
$ ./shuffle one two three
three one two
$ ./shuffle a b c d e f
b a d e f c
```

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

```
int printf(const char *format, ...); // format string %s, char %c, int %d
long random(void); // returns a value between 0 and (2^31) - 1
```

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

4) Gegeben den folgenden Code, implementieren Sie die Funktion *add()*, welche einen neuen Node vorne in die unsortierte Liste *list* hängt und diese zurück gibt. Sowie die Funktion *get()*, welche den Node mit Key *key* in der Liste *list* findet und diesen zurückgibt. Punkte: _ / 12

```
struct node { int key; char value[32]; struct node* next; };

struct node *list = NULL;

struct node *add(struct node *list, int key, char *value);
struct node *get(struct node *list, int key);

int main() {
    list = add(list, 3000, "Bern");
    list = add(list, 4000, "Basel");
    list = add(list, 8000, "Zurich");
    struct node *n = get(list, 4000);
    printf("%d: %s\n", n->key, n->value);
}
```

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

```
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 Fragen-Nr.:

File In-/Output

5) Schreiben Sie ein Programm *abcfile*, das eine per Command Line übergebene Datei (ASCII, nur 'a'-'z') als Input nimmt, und die darin enthaltenen Buchstaben nach Alphabet getrennt in einzelne Files kopiert, bzw. an diese anhängt. Leere Output Files sind erlaubt. Punkte: / 16

```
$ echo "not bad meaning bad but bad meaning good" > my
$ ./abcfile my
$ ls my*
my      my_c  my_f  my_i  my_l  my_o  my_r  my_u  my_x
my_a    my_d  my_g  my_j  my_m  my_p  my_s  my_v  my_y
my_b    my_e  my_h  my_k  my_n  my_q  my_t  my_w  my_z
$ cat my_a
aaaaa$
```

Verwenden Sie dazu die folgenden System Calls, Fehlerbehandlung können Sie weglassen:

```
int sprintf(char *s, char *format, ...); // prints formatted output to s
```

```
size_t strlen(const char *s); // calculate the length of a string
```

```
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 & Fragen-Nr.:

```
// Fortsetzung auf Folgeseite
```

(5) Fortsetzung:

Prozesse und Signale

6) Welche der folgenden Aussagen zu Heap sind korrekt?

Punkte: _ / 4

Zutreffendes ankreuzen:

- | | |
|---|---|
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Der Heap hat den virtuellen Speicher des Prozesses für sich allein. |
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Auf dem Heap allozierte Variablen überdauern Funktionsaufrufe. |
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Heap-Speicher ist unbegrenzt, <i>malloc()</i> alloziert immer mehr. |
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Am oberen Rand des Heaps beginnt jeweils direkt der Stack. |

7) Schreiben Sie ein Programm *ignore*, welches das *SIGINT* Signal genau *n* mal ignoriert,

bevor es terminiert. Der Parameter *n* wird per Command Line übergeben.

Punkte: _ / 8

```
$ ./ignore 3
^C^C^C^C$
```

Verwenden Sie dazu die folgenden System Calls, Fehlerbehandlung können Sie weglassen:

```
int atoi(const char *nptr); // convert a string to an integer
```

```
int pause(void); // Pause causes the calling process to sleep until a  
signal terminates the process or causes invocation of a handler function.
```

```
typedef void (*sighandler_t)(int); e.g. SIGINT = 2, ^C; SIGTSTP = 20, ^Z  
sighandler_t signal(int sig, sighandler_t handler); // set SIG_IGN,  
SIG_DFL, or a programmer-defined function to handle the signal sig.
```

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

Prozess Lebenszyklus

8) Welche dieser Aussagen zu Prozessen sind korrekt?

Punkte: _ / 4

Zutreffendes ankreuzen:

- ☐ Ja | ☐ Nein Ein Prozess kann insgesamt mehr als ein Parent haben.
- ☐ Ja | ☐ Nein Der Parent bekommt von *fork()* die Child Prozess ID.
- ☐ Ja | ☐ Nein Der Child Prozess kann offene Files des Parents lesen.
- ☐ Ja | ☐ Nein Die Funktion *execve()* führt einen neuen Prozess aus.

9) Schreiben Sie ein Programm *domino*, das eine Reihe von *n* Prozessen "aufstellt", die dann in umgekehrter Reihenfolge wieder "umfallen" bzw. terminieren. Der Parameter *n* wird per Command Line übergeben, der Output soll wie im Beispiel aussehen, mit PIDs. Punkte: _ / 12

```
$ ./domino 3
1001 set up
1002 set up
1003 set up
oops...
1003 fallen
1002 fallen
1001 fallen
```

Verwenden Sie dazu die folgenden System Calls, Fehlerbehandlung können Sie weglassen:

<code>int atoi(const char *nptr); // convert a string to an integer</code>
<code>noreturn void exit(int status); // cause normal process termination</code>
<code>pid_t fork(void); // create a child process, returns 0 in child process</code>
<code>pid_t getpid(void); // returns the process ID of the calling process</code>
<code>int printf(const char *format, ...); // format string %s, char %c, int %d</code>
<code>pid_t wait(int *wstatus); // wait for child process to terminate</code>

Fortsetzung auf der nächsten Seite.

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

Threads

10) Schreiben Sie ein Programm *threadrace*, bei dem *n* Threads auf Kommando versuchen, ihren per Command Line erhaltenen Namen, als erstes auf die Konsole auszugeben, wie im Beispiel unten gezeigt. Auf die Taste ENTER kann mit *getch()* gewartet werden. P.kte: _ / 14

Annahme: Implementierung ohne Synchronisation und ohne Condition Variables genügt.

```
$ ./threadrace bat cat dog
Press ENTER to start:
dog
bat
cat
```

Verwenden Sie dazu die folgenden System Calls, Fehlerbehandlung können Sie weglassen:

```
int getchar(void); // blocks until ENTER is pressed, returns a character
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
noreturn void pthread_exit(void *retval); // terminate calling thread
int pthread_detach(pthread_t thread); // detach a thread
int pthread_join(pthread_t thread, void **retval); // retval can be NULL
```

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

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____