

Assessment I

Vorname: _____

Punkte: ____ / 90, Note: ____

Name: _____

Frei lassen für Korrektur.

Klasse: 4ibb2

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 *double* kann man Werte des Typs *int* zuweisen.
- ☐ Ja | ☐ Nein Ein *int* Wert belegt im Speicher des Computers mindestens 16 Byte.
- ☐ Ja | ☐ Nein Der Typ *int* benötigt jeweils ein Byte mehr als der Typ *unsigned int*.
- ☐ Ja | ☐ Nein Der *sizeof* Operator liefert für den Typ *char* immer das Resultat 1.

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

Punkte: _ / 4



Zutreffendes ankreuzen

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

|
|
|
|
|
|

(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, beschreiben Sie, in der Tabelle unten, die drei wichtigsten

Unterschiede der Funktionen *create_point1()* und *create_point2()*.

Punkte: _ / 6

```
#include ... // ignore

struct point {
    int x;
    int y;
};

struct point create_point1(int x, int y) {
    struct point result = { x, y };
    return result;
}

struct point *create_point2(int x, int y) {
    struct point *result = malloc(sizeof(struct point));
    result->x = x;
    result->y = y;
    return result;
}

int main() {
    struct point p = create_point1(0, 0);
    struct point *q = create_point2(0, 0);
}
```

Unterschiede hier eintragen, jeweils beide Seiten des Unterschieds ausformulieren:

create_point1()	create_point2()

File In-/Output

5) Schreiben Sie ein Programm *rotfile*, das den Inhalt einer Datei um *n* Byte nach links rotiert.

Der Filename *file* und der Parameter *n* werden per Command Line übergeben. Punkte: / 12

```
$ echo -n "012345" > my.txt
$ ./rotfile my.txt 3
$ cat my.txt
345012$
```

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

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

```
off_t lseek(int fd, off_t offset, int from); // Position read/write file
offset; from = SEEK_SET, SEEK_CUR or SEEK_END; returns new offset from 0.
```

```
int open(const char *pathname, int flags);
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_RDWR, 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, und auf Zusatzblatt mit Ihrem Namen & Fragen-Nr.:

```
// Fortsetzung auf Folgeseite
```

(5) Fortsetzung:

Prozesse und Signale

6) Welche der folgenden Aussagen zum Stack sind im allgemeinen korrekt? Punkte: _ / 4

Zutreffendes ankreuzen:

- | | |
|---|--|
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Der Stack hat den virtuellen Speicher des Prozesses für sich allein. |
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Auf dem Stack allozierte Variablen überdauern Funktionsaufrufe. |
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Argumente und globale Variablen werden auf dem Stack alloziert. |
| <input type="checkbox"/> Ja <input type="checkbox"/> Nein | Zwischen Stack und Heap befindet sich nicht-allozierter Speicher. |

7) Schreiben Sie ein Programm *stopat*, welches erst beim *n*-ten mal drücken von CTRL-C auf das *SIGINT* Signal reagiert. Der Parameter *n* wird per Command Line übergeben. P.kte: _ / 8

```
$ ./stopat 3  
^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 im allgemeinen korrekt?

Punkte: _ / 4

Zutreffendes ankreuzen:

- ☐ Ja | ☐ Nein Ein Prozess hat zu jedem Zeitpunkt genau einen Parent.
- ☐ Ja | ☐ Nein Der Child Prozess kann mit *wait()* auf den Parent warten.
- ☐ Ja | ☐ Nein Der Child Prozess kann offene Files des Parents schliessen.
- ☐ Ja | ☐ Nein Der *execve()* Call lädt ein neues Programm in den Prozess.

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

```
$ ./domino 3
7001 set up
7002 set up
7003 set up
oops...
7003 tumble
7002 tumble
7001 tumble
```

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.:

10) Schreiben Sie ein kurzes Programm welches beweist, dass bei *fork()* der Speicher mittels *copy-on-write* Semantik kopiert wird. Nutzen Sie dazu die hypothetische Funktion *pframe()*, welche zu einer virtuellen Speicheradresse deren physische Page-Frame ID liefert. P.te: __ / 10

```
int pframe(void *vaddr); // made up; returns the physical page frame ID  
that a virtual address, or rather its virtual memory page, is mapped to
```

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

```
void assert(scalar expression); // abort program if expression is false  
pid_t fork(void); // create a child process, returns 0 in child process
```

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

Threads

11) 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) _____