

Assessment I

Vorname: _____

Punkte: ____ / 90, Note: ____

Name: _____

Frei lassen für Korrektur.

Klasse: 3ib

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 dieser Deklarationen von *main()* sind korrekt in C?

Punkte: _ / 4

Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein `int main(char **argv);`
- ☐ Ja | ☐ Nein `int main(int argc, string argv[]);`
- ☐ Ja | ☐ Nein `int main(int argc, char *argv[]);`
- ☐ Ja | ☐ Nein `int main(char *argv[], int argc);`

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

Punkte: _ / 4



Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein `int i = 3; int *p = &i; int **q = &p; int *r = p;`
- ☐ Ja | ☐ Nein `int i = 3; int *p = &i; int *r = *p; int *q = &p;`
- ☐ Ja | ☐ Nein `int i = 3; int *p = &i; int *q = p; int *r = &i;`
- ☐ Ja | ☐ Nein `int i = 3; int *r = &i; int *p = &i; int **q = &p;`

3) Gegeben die folgenden Structs, welche Statements sind wahr?

Punkte: _ / 4

```
struct r { long a; long b; long c; };
struct s { long n; struct r *p; };
```

Zutreffendes ankreuzen

- ☐ Ja | ☐ Nein `sizeof(struct s) > sizeof(struct r)`
- ☐ Ja | ☐ Nein `sizeof(struct r) == 3 * sizeof(long)`
- ☐ Ja | ☐ Nein `sizeof(struct s) == sizeof(long) + sizeof(struct r)`
- ☐ Ja | ☐ Nein `sizeof(struct r *) < 2 * sizeof(long)`

Funktionen in C

4) Wieso führt das folgende Programm zu einem Segmentation Fault?

Punkte: _ / 6

```
01: void f(int (*g)(void)) {  
02:     int i = g();  
03: }  
04:  
05: int g(void) {  
06:     return 0;  
07: }  
08:  
09: int main(void) {  
10:     f(main);  
11:     return 0;  
12: }
```

Erklärung hier eintragen, Schritt für Schritt ausformulieren:

5) Wie funktioniert ein System Call?

Punkte: _ / 4

Erklärung hier eintragen, vier wesentliche Punkte ausformulieren:

6) 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
```

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

```
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 Fragen-Nr.:

File In-/Output

7) Schreiben Sie ein Programm *patch*, das ein existierendes File *orig* an der Position *n* mit

Bytes des existierenden Files *delta* der Länge *m* überschreibt, wie hier gezeigt. Punkte: / 12

```
$ echo "hello?" > orig
$ echo "ipp" > delta
$ ./patch orig 1 delta 3
$ cat orig
hippo?
```

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); // Opens the file specified by
pathname. Returns the file descriptor. Flags include O_RDONLY, O_WRONLY.
```

```
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.:

8) Schreiben Sie ein Programm *fdb*, das ein File *db* als "Array DB" verwaltet, für Strings < 32 Byte, die mit *put* und *get* am Index *i* schreib- bzw. lesbar sind, wie hier gezeigt. Punkte: / 12

```
$ ./fdb my.db put 3 hello
$ ./fdb my.db get 3
hello
```

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.
Allows file offset to be set beyond the end of the file, fills with '\0'
```

```
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
```

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

```
size_t strlen(const char *s); // calculate length of a string, excl. '\0'
```

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

Prozesse und Signale

9) Schreiben Sie ein Programm *count*, das ausgibt, wie weit ein Prozess in 1 Sekunde zählen

kann. Nutzen Sie dazu die *alarm()* Funktion, die das SIGALRM Signal auslöst. Punkte: _ / 12

```
$ ./count
counted to 75654776
```

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

```
int alarm(int sec); // arranges for a SIGALRM signal to be delivered to
the calling process in sec seconds; returns remaining sec of prev. alarm
```

```
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 handler); // set SIG_IGN,
SIG_DFL, or a programmer-defined function to handle the signal sig
```

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

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

Prozess Lebenszyklus

10) Schreiben Sie ein Programm *exec*, das ein per Command Line gegebenes Programm in *n* parallelen Child Prozessen ausführt, wie hier gezeigt, und auf deren Ende wartet. P.kte: _ / 12

```
$ ./exec 3 hello
Hello, World!
Hello, World!
Hello, World!
```

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

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

pid_t fork(void); // create a child process, returns 0 in child process

int execve(const char *pathname, char *const argv[], char *const envp[]);
// executes the program referred to by pathname; argv, envp can be NULL

void exit(int status); // cause normal process termination

pid_t wait(int *wstatus); // wait for child process to terminate; returns
the process ID of the terminated child or -1 if no child left to wait for
```

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

Threads

11) Was ist der Output dieses Programms, und wieso?

Punkte: _ / 8

```
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t c = PTHREAD_COND_INITIALIZER;

void *start(void *arg) {
    pthread_mutex_lock(&m);
    printf("A\n");
    pthread_cond_signal(&c);
    pthread_mutex_unlock(&m);
}

int main() {
    printf("B\n");
    pthread_t thread;
    pthread_mutex_lock(&m);
    pthread_create(&thread, NULL, start, NULL);
    printf("C\n");
    pthread_cond_wait(&c, &m);
    printf("D\n");
    pthread_mutex_unlock(&m);
}
```

Output und Begründung hier eintragen; Annahme: #includes sind vorhanden:

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____