Fachhochschule Nordwestschweiz
Hochschule für Technik

System-Programmierung (syspr), 05. November 2018, thomas.amberg@fhnw.ch

Assessment I

Vorname: _____ Punkte: ____ / 120, Note: ____

Name: _____ *Frei lassen für Korrektur.*

Klasse: ☐ Klasse 3ia ☐ Klasse 3ib

Hilfsmittel:

- Eine (mehrseitige) 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 Response: ☐ *Ja* oder ☐ *Nein* ankreuzen, +1/-1 Punkt pro richtige/falsche Antwort, d.h. Antworten "raten" lohnt sich nicht (pro Frage gibt es aber nie weniger als `0` Punkte).
- Multiple Choice: Eine ☒ *Antwort* pro Frage ankreuzen, *4* Punkte pro richtige Antwort.
- Offene Fragen: Max. *12* Punkte pro Frage für Korrektheit, Vollständigkeit und Kürze.
- Programme: Max. *18* Punkte pro Frage für Idee, Umsetzung und Code-Struktur.

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(void);`

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

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

2) Welche dieser Konzepte sind eingebaut in die Sprache C? Punkte: ___ / 6

Zutreffendes ankreuzen; C = C89 oder C99:

☐ Ja | ☐ Nein	Typ *string*	☐ Ja | ☐ Nein	Funktions-Pointer
☐ Ja | ☐ Nein	Type Cast	☐ Ja | ☐ Nein	Objektorientierung
☐ Ja | ☐ Nein	Funktionen	☐ Ja | ☐ Nein	Garbage Collection

3) Welche Ausdrücke liefern die Grösse des Typs *int* in C? Punkte: ___ / 4

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein	*size_t*	☐ Ja | ☐ Nein	*MAX_INT*
☐ Ja | ☐ Nein	*sizeof(int)*	☐ Ja | ☐ Nein	*int.size()*

4) Gegeben den folgenden Code, wie kommt man zu *a[2]* mittels *p*? Punkte: ___ / 4

```
int a[] = { 3, 1, 4 };
int *p = a;
```

Eine Antwort (von 4) ankreuzen:

☐ `*(p + 2 * sizeof(int));`	☐ `*(p + 2);`
☐ `&(p + 2 * sizeof(int));`	☐ `&(p + 2);`

Funktionen in C

5) Schreiben Sie ein Programm, das seinen Namen ausgibt, ohne Pfad: Punkte: ___ / 18

```
$ ./name
name
$ mv name /tmp/new_name
$ /tmp/new_name
new_name
```

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

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

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

6) Gegeben den folgenden Code, welchen Wert hat *n* nach Aufruf von *x()*? Punkte: ____ / 12

```
int n = 3;

void f() { n++; }

void g(int n) { n++; }

void h(int *n) { n++; }

void x() {
    f();
    g(n);
    h(&n);
}
```

Resultat und Begründung hier eintragen:

```
n =
```

7) Gegeben den folgenden Code, welche Aufrufe von *map* sind korrekt? Punkte: ____ / 4

```
int add(int a, int b) { return a + b; }
int inc(int i) { return i + 1; }
void map(int a, int b, int (*op)(int, int)) { op(a, b); }
```

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein `map(1, 2, add(3, 4));`

☐ Ja | ☐ Nein `map(1, 2, add);`

☐ Ja | ☐ Nein `map(1, 2, inc);`

☐ Ja | ☐ Nein `map({1, 2}, 2, inc);`

File In-/Output

8) Welche Flags braucht *open()* für ein (altes oder neues) shared Log-File? Punkte: ___ / 6

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein	`O_CREAT`	☐ Ja | ☐ Nein	`O_TRUNC`
☐ Ja | ☐ Nein	`O_EXCL`	☐ Ja | ☐ Nein	`O_WRONLY`
☐ Ja | ☐ Nein	`O_APPEND`	☐ Ja | ☐ Nein	`O_DIRECTORY`

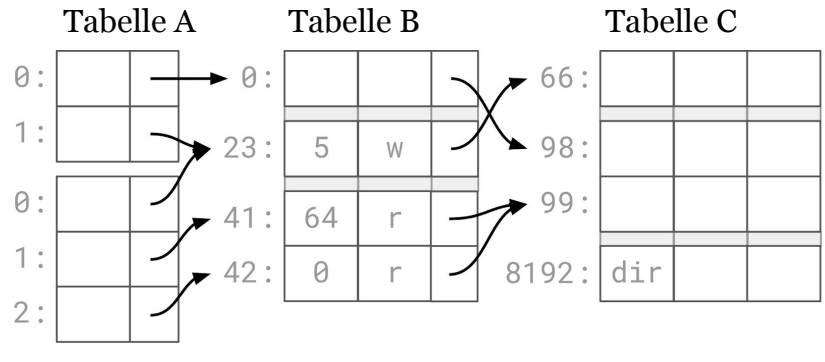
9) Gegeben den folgenden Call, wie behandelt man Fehler bei *open()*? Punkte: ___ / 4

```
#include <errno.h> …
int fd = open("my.txt", O_RDONLY);
```

Eine Antwort (von 4) ankreuzen; nehmen Sie an, h() macht den Rest:

☐ `if (fd == NULL) { h(errno); }`	☐ `if (errno != 0) { h(errno); }`
☐ `if (fd == errno) { h(errno); }`	☐ `if (fd == -1) { h(errno); }`

10) Gegeben das folgende Bild, was steht in Tabelle A, B und C? Punkte: ___ / 4



Eine Antwort ankreuzen:

☐ A = i-Nodes, B = File Deskriptoren, C = Offene Files

☐ A = File Deskriptoren, B = Offene Files, C = i-Nodes

☐ A = Offene Files, B = i-Nodes, C = File Deskriptoren

☐ A = File Deskriptoren, B = i-Nodes, C = Offene Files

Prozesse und Signale

11) Welche Struktur bilden Prozess IDs im Allgemeinen? Punkte: ___ / 4

Eine Antwort (von 4) ankreuzen:

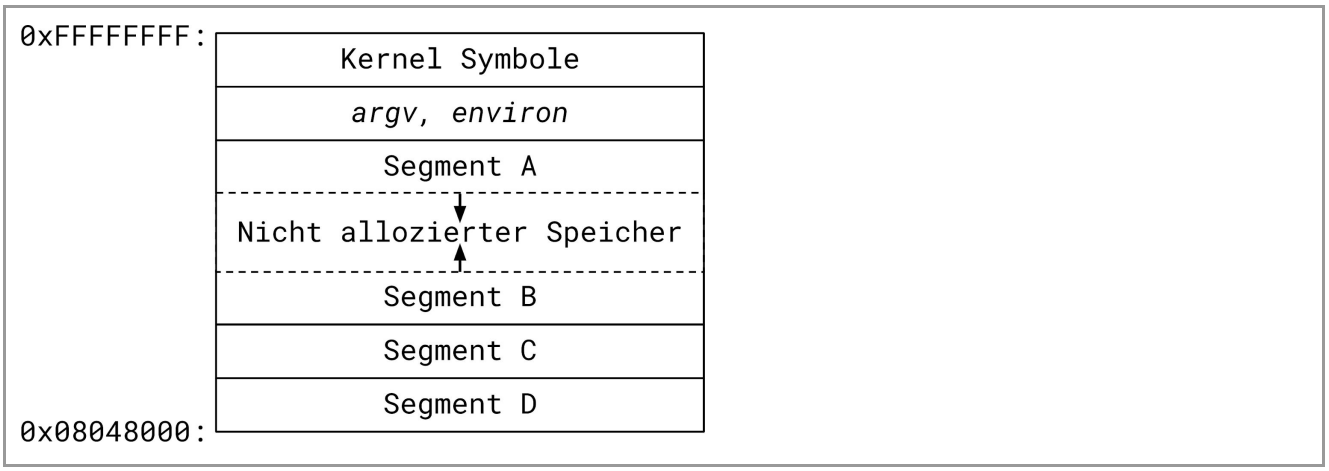
☐ Listen-Struktur

☐ Kreis-Struktur

☐ Tabellen-Struktur

☐ Baum-Struktur

12) Gegeben das folgende Bild, wie heissen die Segmente A, B, C und D? Punkte: ___ / 4



Eine Antwort ankreuzen:

☐ A = Daten (init. & bss), B = Programm-Text, C = Stack, D = Heap

☐ A = Heap, B = Stack, C = Daten (init. & bss), D = Programm-Text

☐ A = Stack, B = Daten (init. & bss), C = Programm-Text, D = Heap

☐ A = Stack, B = Heap, C = Daten (init. & bss), D = Programm-Text

13) Welche Vorteile hat virtueller Speicher? Punkte: ___ / 4

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein File-Input muss nicht auf Disk geschrieben werden.

☐ Ja | ☐ Nein Programm-Text kann (read-only) geshared werden.

☐ Ja | ☐ Nein Programmierer muss physisches Layout nicht kennen.

☐ Ja | ☐ Nein Prozess, bzw. ausgeführtes Programm hat CPU für sich.

14) Gegeben folgende Doku, welche der Programme darunter sind korrekt? Punkte: ___ / 4

```
void *malloc(size_t size);

// The malloc() function allocates size bytes and returns a pointer to
the allocated memory.
```

```
void free(void *ptr);

// The free() function frees the memory space pointed to by ptr, which
must have been returned by a previous call to malloc().
```

Zutreffendes ankreuzen; nehmen Sie an #includes, main() und *… sind vorhanden:*

☐ Ja | ☐ Nein `int *i = malloc(1); *i = …; free(i);`

☐ Ja | ☐ Nein `char *c = malloc(4); *c = …; free(c);`

☐ Ja | ☐ Nein `int *j = malloc(sizeof(int)); *j = …; free(j);`

☐ Ja | ☐ Nein `int *k = malloc(3 * sizeof(int)); *k = …; free(k + 3);`

15) Welche der folgenden Aufrufe senden ein Signal an einen Prozess? Punkte: ___ / 4

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein *signal()*

☐ Ja | ☐ Nein *raise()*

☐ Ja | ☐ Nein *kill()*

☐ Ja | ☐ Nein *wait()*

Prozess-Lebenszyklus

16) Mit welchen dieser Aufrufe kann ein Prozess sich selbst beenden? Punkte: ___ / 4

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein *atexit()*

☐ Ja | ☐ Nein *_exit()*

☐ Ja | ☐ Nein *exit()*

☐ Ja | ☐ Nein *return* in *main()*

17) Schreiben Sie ein Programm, das genau zwei Child Prozesse startet: Punkte: ___ / 18

```
$ ./fork2
parent pid = 21
child pid = 22
child pid = 23
```

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

```
void exit(int status);
```

```
pid_t fork(void);

// create a child process,
// returns child PID in parent,
// and 0 in child process
```

```
pid_t getpid(void); // process PID
pid_t getppid(void); // parent PID
```

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

```
pid_t wait(int *status);
```

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

Threads und Synchronisation

18) Was ist der Output dieses Programms, und wieso? Punkte: ___ / 12

```
void *start(void *arg) {
    pthread_t id = *((pthread_t *) arg);
    pthread_join(id, NULL);
    printf("joined %lu\n", id);
    exit(0);
}

int main() {
    size_t n = sizeof(pthread_t);
    pthread_t *arg = malloc(n);
    *arg = pthread_self();
    pthread_t thread;
    pthread_create(&thread, NULL, start, arg);
    printf("created %lu\n", thread);
    pthread_exit(NULL);
}
```

Resultat und Begründung hier eintragen; nehmen Sie an #includes sind vorhanden:

```
$ gcc -pthread -o program program.c
$ ./program
```