Fachhochschule Nordwestschweiz
Hochschule für Technik

System-Programmierung (syspr) 05. November 2024 thomas.amberg@fhnw.ch

Assessment I

Vorname: _____ Punkte: ___ / 90, Note: ___

Name: _____ *Frei lassen für Korrektur.*

Klasse: ~~3ib~~ 3ia

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 dieser Deklarationen von *main()* sind korrekt in C? Punkte: _ / 4

Zutreffendes ankreuzen, C = C99

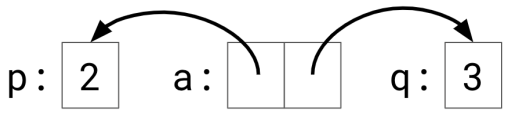
☐ Ja | ☐ Nein `void main(char **argv);`

☐ 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 p = 2; int q = 0; int *a[] = {&p, &q}; *a[1] = 3;`

☐ Ja | ☐ Nein `int p; int q = 2; int *a[] = {0}; p = q; a[0] = &p; q++;`

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

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

3) Gegeben die folgenden Structs, welche Statements sind korrekt? 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) Schreiben Sie ein Programm *shuffle*, welches $n > 0$ übergebene Command Line Argumente in zufälliger Reihenfolge auf die Konsole ausgibt, wie hier im Beispiel. Punkte: _ / 12

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

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

```
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 Frage-Nr.:

5) Wie funktioniert ein System Call? Punkte: _ / 6

Erklärung hier eintragen, sechs wesentliche Punkte, je ein kurzer Satz:

File In-/Output

6) Schreiben Sie ein Programm *tee*, welches Bytes von *STDIN_FILENO* liest und diese sowohl auf *STDOUT_FILENO* schreibt, als auch in eine Datei *dest* kopiert. Zudem soll das Programm anzeigen, wie es benutzt werden muss, falls es ohne Argumente aufgerufen wird. P.kte: _ / 12

```
$ ./tee
usage: ./tee dest
$ echo "hello" | ./tee log.txt
hello
$ cat log.txt
hello
```

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.
```

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

```
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.
```

Fortsetzung auf der nächsten Seite.

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

Prozesse und Signale

7) Gegeben das folgende Programm, welcher (Schluss-) Output ist möglich? Punkte: _ / 4

```
01 int main(void) {
02     int res = fork();
03     if (res == 0) {
04         printf("c\n");
05     } else {
06         wait(NULL);
07     }
08     printf("p\n");
09 }
```

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein `p\nc\np\n`

☐ Ja | ☐ Nein `c\np\nc\n`

☐ Ja | ☐ Nein `c\np\np\n`

☐ Ja | ☐ Nein `p\nc\n`

8) Schreiben Sie ein Programm *sigseq*, welches als Argument eine Folge von Signal IDs < 32 nimmt und nur dann terminiert, wenn es genau diese Signalfolge empfangen hat. P.kte: _ / 12

```
$ ./sigseq 2 2 20 2
^C^C^Z^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

9) Schreiben Sie ein Programm *filepos*, das durch Ausgabe der PIDs und der Offsets im File beweist, dass ein existierendes, vom Parent geöffnetes und benutztes File, nach einem fork() auch im Child Prozess zugänglich ist, und an derselben Offset Position steht. Punkte: _ / 10

```
$ echo "hello" > my.txt
$ ./filepos my.txt
777: 1
778: 1
```

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

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

```
pid_t getpid(void); // returns the process ID of the calling process
```

```
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.
```

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

```
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.
```

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

10) 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 Child Prozess terminiert immer vor dem Parent.

☐ Ja | ☐ Nein Der Parent bekommt von *fork()* die Child Prozess ID.

☐ Ja | ☐ Nein Die Funktion *execve()* führt einen neuen Prozess aus.

Threads und Synchronisation

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
```

|

Fortsetzung auf Folgeseite

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

12) Was sind zwei wesentliche Merkmale einer *Critical Section*? Punkte: _ / 4

Merkmale hier eintragen, jeweils einen ganzen Satz ausformulieren:

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____