Fachhochschule Nordwestschweiz
Hochschule für Technik

System-Programmierung (syspr) 07. November 2023 thomas.amberg@fhnw.ch

Assessment I

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

Name: _____ *Frei lassen für Korrektur.*

Klasse: 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 Typen sind Teil der Sprache C? Punkte: _ / 4

Zutreffendes ankreuzen, C = C99:

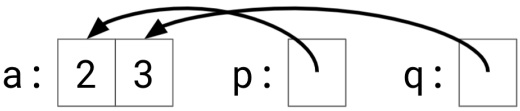
☐ Ja | ☐ Nein `byte`

☐ Ja | ☐ Nein `pid_t`

☐ Ja | ☐ Nein `string`

☐ Ja | ☐ Nein `long`

2) Welche Abfolge von Statements führt zu folgender Situation im Speicher? Punkte: _ / 4



Zutreffendes ankreuzen

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

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

☐ Ja | ☐ Nein
```
int a[2]; int *p = a + 1; *p = 2; int *q = p++; *q = 3;
```

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

Aufgabe 3 ist auf der nächsten Seite.

Funktionen in C

3) Gegeben den folgenden Code, implementieren Sie die Funktion *add()*, welche einen neuen Node vorne in die unsortierte Liste *list* hängt und diese zurückgibt. 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);
}
```

Verwenden Sie die folgenden Sys. Calls; #includes, 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 Frage-Nr.:

4) Schreiben Sie ein Programm *vowels*, das die Vokale (a, e, i, o, u) zählt, welche im jeweiligen Wort vorkommen. Worte können Buchstaben (a-z) enthalten und werden per Command Line übergeben. Die Ausgabe soll pro Wort und Vokal erfolgen, wie hier im Beispiel. Punkte: _ /12

```
$ ./vowels igel hase uhu
a:0, e:1, i:1, o:0, u:0
a:1, e:1, i:0, o:0, u:0
a:0, e:0, i:0, o:0, u:2
```

Verwenden Sie die folgenden Sys. Calls; #includes, 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 Frage-Nr.:

File In-/Output

5) Schreiben Sie ein Programm *reverse*, das mittels *lseek()* eine per Command Line gegebene, beliebig grosse, existierende Datei, Byte-weise umkehrt, wie hier im Beispiel. Punkte: _ / 18

```
$ echo -n "hello" > my.txt
$ ./reverse my.txt
$ cat my.txt
olleh$
```

Verwenden Sie die folgenden Sys. Calls; #includes, Fehlerbehandlung können Sie weglassen:

```
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: O_RDWR, 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 und Frage-Nr.:

Prozesse und Signale

6) Gegeben, das folgende Programm, welcher Output ist auf Linux möglich? Punkte: _ / 4

```
int main(void) {
    if (fork()) {
        fork();
        printf("a\n");
        exit(0);
    }
    printf("b\n");
}
```

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein `a\na\nb\n`

☐ Ja | ☐ Nein `a\nb\nb\n`

☐ Ja | ☐ Nein `a\nb\na\n`

☐ Ja | ☐ Nein `b\na\na\n`

7) Schreiben Sie ein Programm *sig*, das auf das erste Signal mit *int* Wert von *1* bis und mit *32* wartet, und diesen mit Beschreibung in der *main()* Funktion mit *printf()* ausgibt. P.kte: _ / 10

```
$ ./sig
^C2, Interrupt
$ ./sig
^Z20, Stopped
```

Verwenden Sie die folgenden Sys. Calls; #includes, Fehlerbehandlung können Sie weglassen:

```
int pause(void); // sleep until a signal causes invocation of a handler
```

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

```
typedef void (*sighandler_t)(int); e.g. SIGINT = 2, ^C; SIGTSTP = 20, ^Z
sighandler_t signal(int sig, sighandler_t h); // set h to handle a signal
```

```
char *strsignal(int sig); // return a string describing the signal
```

Fortsetzung auf der nächsten Seite.

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

Prozess Lebenszyklus

8) Das Programm *fib* soll die *Fibonacci-Zahl* für ein per Command Line gegebenes *n* rekursiv berechnen, indem das Programm selbst erneut aufgerufen wird. Implementieren Sie (nur) die Funktion *f()*, die den (Programm-)Aufruf ausführt und das Resultat zurückgibt. Punkte: _ / 12

```
#include ... // ignore

char *name;

int f(int n); // TODO: implement f() using fork(), exec(), wait()

int main(int argc, char *argv[]) {
    name = argv[0];
    int n = atoi(argv[1]);
    if (n < 3) {
        return 1;
    } else {
        return f(n - 1) + f(n - 2);
    }
}
```

Fortsetzung auf der nächsten Seite.

Hinweis: Das Resultat des Programms fib wird mittels echo und $? gelesen, wie hier gezeigt.

```
$ ./fib 3
$ echo $? # show exit status of the above command
2
```

Verwenden Sie die folgenden Sys. Calls; #includes, Fehlerbehandlung können Sie weglassen:

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

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

```
pid_t wait(int *status); // wait for child process to terminate; returns
the PID of the terminated child or -1 if no child is left to wait for;
the macro WEXITSTATUS(status) returns the exit status of the child.
```

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

Threads und Synchronisation

9) Gegeben diesen Code, der eine Post mit drei Schaltern und einem Warte-Ticket Dispenser simuliert, implementieren Sie die Funktionen *queue()* für Kunden die ein Ticket nehmen und warten, sowie *serve()* für die Schalter, die das jeweils nächste Ticket bedienen. Punkte: _ / 14

Hinweise: Reiner Lesezugriff ist hier ohne Mutex möglich. Die Simulation läuft ohne Output.

```
#include ... // ignore

struct count {
    pthread_mutex_t m;
    volatile int n;
};

struct count ticket = { PTHREAD_MUTEX_INITIALIZER, 0 };
struct count served = { PTHREAD_MUTEX_INITIALIZER, 0 };

void *queue(void *arg); // TODO: implement taking a ticket, waiting
void *serve(void *arg); // TODO: implement serving clients, forever

int main() {
    for (int i = 0; i < 3; i++) {
        pthread_t server;
        pthread_create(&server, NULL, serve, NULL);
        pthread_detach(server);
    }
    while (1) { // forever
        pthread_t client;
        pthread_create(&client, NULL, queue, NULL);
        pthread_detach(client);
    }
}
```

Verwenden Sie die folgenden Sys. Calls; #includes, Fehlerbehandlung können Sie weglassen:

```
int pthread_mutex_lock(pthread_mutex_t *mutex); // lock a mutex, returns
0 on success; If the mutex object is already locked by another thread,
the calling thread shall block until the mutex becomes available.
```

```
int pthread_mutex_unlock(pthread_mutex_t *mutex); // unlock a mutex,
returns 0 on success; If there are threads blocked on the mutex object,
scheduling policy shall determine which thread shall acquire the mutex.
```

Fortsetzung auf der nächsten Seite.

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

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____