

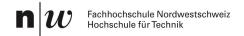
System-Programmierung (syspr) 5. November 2019

thomas.amberg@fhnw.ch

Assessment I

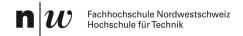
Vorname:	Punkte: / 92, Note:
Name:	Frei lassen für Korrektur.
Klasse: 3ia	
Hilfsmittel:	
- Die vom Dozent ausgeteilte C Referenzkar	rte.
- Lösen Sie die Aufgaben direkt auf den Prü	ifungsblättern.
- Zusatzblätter, falls nötig, mit Ihrem Name	en und Fragen-Nr. auf jedem Blatt.
Nicht erlaubt:	
- Unterlagen (Slides, Bücher,).	
- Computer (Laptop, Smartphone,).	
- Kommunikation mit anderen Personen.	
Bewertung:	
- Multiple Response: \square Ja oder \square $Nein$ an	kreuzen, +1/-1 Punkt pro richtige/falsche Antwort,
beide nicht ankreuzen ergibt +0 Punkte;	Гotal pro Frage gibt es nie weniger als 0 Punkte.
- Offene Fragen: Bewertet wird Korrektheit	, Vollständigkeit und Kürze der Antwort.
- Programme: Bewertet wird die Skizze/Ide	ee und Umsetzung des Programms.
Fragen zur Prüfung:	
- Während der Prüfung werden vom Dozen	t 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 Au	sdrücke kompilieren fehlei	rfrei in C?	Punkte: / 4
Zutreffendes ankrei	ızen; C = C99:		
□ Ja □ Nein	int i = malloc(sizeo	of(int));	
□ Ja □ Nein	char s[] = {'o', 'o'	, 'p', 's'};	
□ Ja □ Nein	<pre>byte *b = "hello";</pre>		
□ Ja □ Nein	int j = (0 1);		
2) Was sind drei Un	terschiede der Variablen x	und y im folgenden Code?	Punkte: / 6
<pre>void g(int *p)</pre>	{ }		
<pre>void f() { int *x = ma g(x);</pre>	lloc(sizeof(int));		
int y; g(&y); }			
Unterschiede hier et	intragen, jeweils beide Sei	ten des Unterschieds ausformı	ılieren:



Funktionen in C

3) Schreiben Sie ein Programm, das seine Argumente rückwärts ausgibt: Punkte: / 12
\$./my_program just a test tset a tsuj
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.:

4)	Gegeben	den folgender	Code, welche	n Wert hat <i>k</i> nach	Aufruf von f()?	Punkte: / 2
----	---------	---------------	--------------	--------------------------	-----------------	-------------

```
int f(int *m, int n) {
   return (*m + 1) * n;
}

int main() {
   int i[] = {3, 5};
   int j = 2;
   int k = f(i, j);
}
```

Resultat und Begründung hier eintragen:

5) Gegeben den folgenden Code, welche Aufrufe von *eval()* sind erlaubt? Punkte: ____ / 4

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

Zutreffendes ankreuzen:



6) Gegeben die Listen-Datenstruktur (unten), implementieren Sie *insert()* um für den String *item* einen Node zu allozieren und alphabetisch sortiert in die Liste einzufügen. Punkte: _ / 12

```
typedef struct node {
    struct node *next;
    char *item;
} Node;
static Node *list = NULL;
void insert(char *item); // TODO: implement
```

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

```
void *malloc(size_t size); // Allocates size bytes and returns a pointer
to the allocated memory.

int strcmp(char *s1, char *s2); // Compares strings s1, s2 alphabetically
and returns an integer less than (s1<s2), equal to, or greater than zero.</pre>
```

File In-/Output

7) Gegeben eine ASCII Datei im Comma Separated Value (CSV) Format, z.B.: Punkte: _ / 16

```
$ cat table.csv
Animal, Size\n
Dog, Big\n
Duck, Small\n
Mouse, Tiny\n
Horse, Huge\n
```

Schreiben Sie ein Programm, das eine Spalte der Form Titel Wert ... an die Tabelle anhängt.

Annahme 1: Der Name der Output Datei wird als Command Line Parameter übergeben.

Annahme 2: Die Anzahl Elemente in der Spalte passt zur Anzahl Zeilen in der Tabelle.

```
$ ./my_program table.csv table2.csv Legs 4 2 4 4

$ cat table2.csv
Animal, Size, Legs\n
Dog, Big, 4\n
Duck, Small, 2\n
Mouse, Tiny, 4\n
Horse, Huge, 4\n
```

Nutzen Sie dazu diese System-Calls, #includes und Fehlerbehandlung können Sie weglassen:

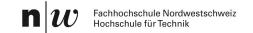
```
int close(int fd); // Closes the file descriptor fd.

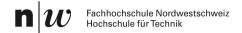
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode); // Opens the file specified by pathname. Or creates it if O_CREAT is used. Returns the file descriptor. Flags include O_APPEND, O_CREAT, O_RDONLY, O_WRONLY, O_RDWR.
Modes, which are used together with O_CREAT 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.
```

(Fortsetzung auf der nächsten Seite)





Prozesse und Signale

8) Schreiben Sie ein Programm, das einen Zombie-Prozess erzeugt, für 1 Sekunde. P.: ___ / 8

```
$ ./my_program &
[1] 1001
$ ps aux | grep my_program
... 1001 ... ./my_program
... 1002 ... [my_program] <defunct> // Zombie Prozess
```

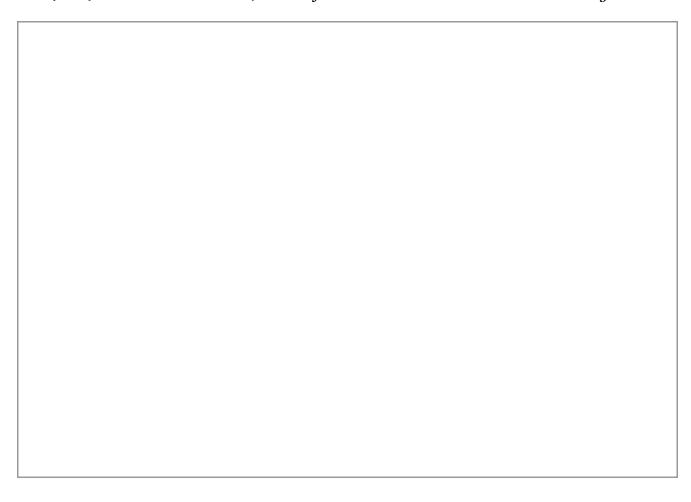
Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
void exit(int status); // Cause normal process termination.

pid_t fork(void); // Create a child process; returns 0 in child process.

int sleep(unsigned int seconds); // Causes the calling thread to sleep.

pid_t wait(int *wstatus); // Wait for child process; wstatus can be NULL.
```





9) Schreiben Sie ein Programm das beweist, dass beim Beenden eines Child Prozesses jeweils ein SIGCHLD Signal an den Parent Prozess gesendet wird.

Punkte: ___ / 9

```
void exit(int status); // Cause normal process termination.

pid_t fork(void); // Create a child process; returns 0 in child process.

int pause(void); // Pause causes the calling process to sleep until a signal terminates the process or causes invocation of a handler function.

int printf(const char *format, ...); // format string %s, char %c, int %d typedef void (*sighandler_t)(int); sighandler_t signal(int signum, sighandler_t handler); // set SIG_IGN, SIG_DFL, or a programmer-defined function to handle the signal signum.
```



Prozess-Lebenszyklus

10) Schreiben Sie ein Programm, dass sich se	lbst genau einmal erneut startet.	Punkte: /	9

```
$ ./my_program
main
main
```

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

```
int execve(const char *pathname, char *const argv[], char *const envp[]);
// execute a program; argument vector and environment are passed to main

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

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

pid_t wait(int *wstatus); // wait for child process; wstatus can be NULL
```

Threads und Synchronisation

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

```
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.

Punkte: ____ / 8



Zusatzblatt zu Aufgabe Nr	_ von (Name)	