

System-Programmierung (syspr) 07. Januar 2019

thomas.amberg@fhnw.ch

Assessment II

Vorname:	Punkte:/ 90, Note:
Name:	Frei lassen für Korrektur.
Klasse: 3ia	
Hilfsmittel:	
- Die vom Dozent ausgeteilte C Referenzkarte	
- Lösen Sie die Aufgaben direkt auf den Prüfu	ngsblättern.
- 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:	
- Offene Fragen: Bewertet wird Korrektheit, V	ollständigkeit und Kürze der Antwort.
- Programme: Bewertet wird die Skizze/Idee	und Umsetzung des Programms.
Fragen zur Prüfung:	
- Während der Prüfung werden vom Dozent l	teine Fragen zur Prüfung beantwortet.

- Ist etwas unklar, machen Sie eine Annahme und notieren Sie diese auf der Prüfung.

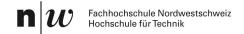
Threads & Synchronisation

1) Programmieren Sie "sleep sort", ein Algorithmus, der Zahlen sortiert, indem er für jede Zahl n in einem eigenen Thread sleep(n) aufruft, und dann die Zahl ausgibt. Punkte: ___ / 16

```
$ ./my_sleep_sort 4 2 1 5
1 2 4 5
```

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

```
int atoi(const char *nptr); // convert a string to an integer
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
int pthread_join(pthread_t thread, void **retval); // retval = NULL
int sleep(int seconds); // calling thread sleeps for a number of seconds
```



IPC mit Pipes

2) Schreiben Sie ein Programm, das eine Pipe vom Enkel zum Grandparent Prozess erstellt.

Der Enkel soll die Nachricht "hello" zum Grandparent schicken, über die Pipe. Punkte: _ / 12

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

```
int close(int fd); // close a file descriptor
pid_t fork(void); // create a child process, returns 0 in child process
int pipe(int pipe_fd[2]); // create pipe, from pipe_fd[1] to pipe_fd[0]
ssize_t read(int fd, void *buf, size_t count); // read from a file descr.
pid_t wait(int *status); // wait for child process, status can be NULL
ssize_t write(int fd, const void *buf, size_t count); // write to a file
```

Sockets

3) Gegeben den folgenden Code, implementieren Sie in main() einen HTTP Server, der Web

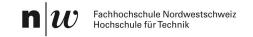
Requests (Länge < MAX_REQ_LEN) liest und auf STDOUT_FILENO ausgibt. Punkte: _ / 18

```
#define BACKLOG 5
#define MAX REO LEN 4096
struct sockaddr_in addr;
char request_buf[MAX_REQ_LEN];
char response_buf[] =
    "HTTP/1.1 200 OK\r\n"
    "Connection: close\r\n"
    "Content-Length: 0\r\n"
    "\r\n";
void initAddr() {
    char *host = "0.0.0.0"; // any
    int port = 8080;
    struct in_addr ip_addr;
    inet_pton(AF_INET, host, &ip_addr);
    size_t addr_size = sizeof(struct sockaddr_in);
    memset(&addr, 0, addr_size);
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr = ip_addr;
}
// TODO: implement int main() { ... }
```

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

```
int accept(int sock_fd, struct sockaddr *addr, socklen_t *addrlen); //
accept a connection on a socket, returns a (client) socket descriptor
int bind(int sock_fd, const struct sockaddr *addr, size_t addr_size); //
bind a local address to a socket, returns 0 on success
int close(int fd); // close a file descriptor, returns 0 on success
int listen(int sock_fd, int backlog); // listen for connections on a
   socket, backlog = number of pending connections, returns 0 on success
ssize_t read(int fd, void *buf, size_t count); // read from a file descr.
int socket(int domain, int type, int pr); // domain = AF_UNIX or AF_INET,
   type = SOCK_STREAM or SOCK_DGRAM, pr = 0, returns a socket descriptor
ssize_t write(int fd, const void *buf, size_t count); // write to a file
```

(Fortsetzung auf der nächsten Seite)

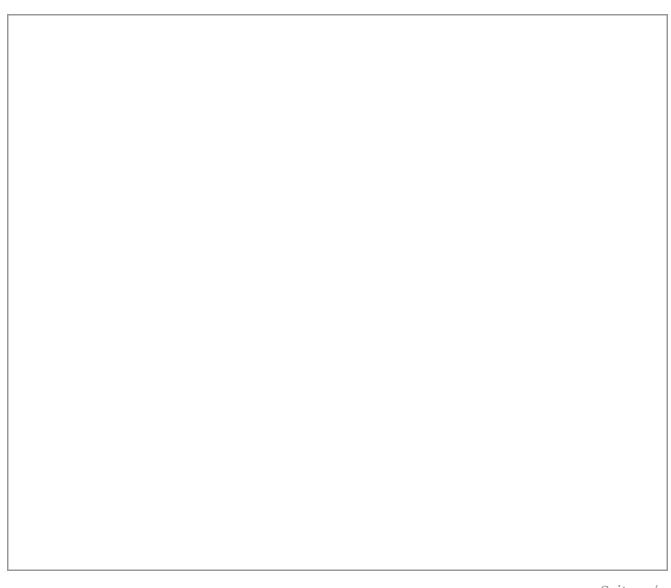


4) Welches Problem wird durch das Fes	stlegen einer <i>Networ</i>	<i>k Byte Order</i> gelös	st? Punkte: _ / 3
Terminals			
5) Welcher Terminal-Mode wird von de	er Shell verwendet, u	and wieso?	Punkte: _ / 3
POSIX IPC			
6) Nennen Sie zwei Unterschiede zwisch	hen <i>named</i> und <i>unn</i>	amed Semanhorer	Punkte / A
o) Ivelinen ble zwei ontersemede zwise.	inen namea una ann		1. 1 unkte / 4
Named POSIX Semaphore	Unnamed P	POSIX Semaphore	



7) Schreiben Sie ein Programm, das immer n Threads hat, die jeweils zufällig nach m = rand() Sekunden terminieren. Ein Sempahor soll die Anzahl Threads konstant halten. Punkte: _ / 18 Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
void free(void *ptr); // free memory allocated on the heap
void *malloc(size_t size); // allocate size bytes of memory on the heap
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
int pthread_detach(pthread_t thread); // detach a thread, no join needed
int rand(); // returns a random number [0, RAND_MAX]
int sem_init(sem_t *sem, int pshared, unsigned int value); // pshared = 0
int sem_post(sem_t *sem); // increment/unlock a semaphore
int sem_wait(sem_t *sem); // decrement/lock a semaphore, blocking if <= 0
int sleep(int seconds); // calling thread sleeps for a number of seconds</pre>
```



Zeitmessung

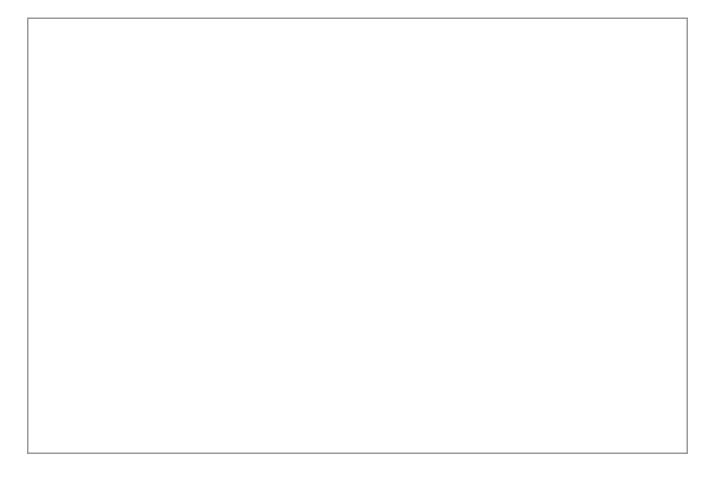
8) Schreiben Sie ein Programm das misst, wie lange es dauert, eine 1-Byte Message durch eine POSIX Message Queue mit Namen "/mq" an sich selbst zu schicken.

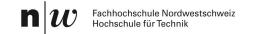
Punkte: _ / 16

```
$ ./my_mq_msg_time
real = 47000 nsec
```

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

```
int clock_gettime(clockid_t c_id, struct timespec *s); // returns 0 or -1
    // c_id = CLOCK_REALTIME or CLOCK_MONOTONIC or CLOCK_PROCESS_CPUTIME_ID
    // struct timespec { time_t tv_sec; long tv_nsec; }; // time in s, nano
mqd_t mq_open(char *name, int flag, mode_t mode, struct mq_attr *attr);
    // open a message queue, flag = O_RDWR|O_CREAT, mode = S_IRUSR|S_IWUSR,
    // struct mq_attr { long mq_maxmsg; long mq_msgsize } // mq attributes
ssize_t mq_receive(mqd_t mq, char *msg, size_t msg_len, int *msg_prio);
int mq_send(mqd_t mq, char *msg, size_t msg_len, int msg_prio);
int mq_unlink(char *name); // remove a message queue
int printf(const char *frmt, ...); // frmt int %d, long %ld, string % ...
```





Zusatzblatt zu Aufgabe Nr	_ von (Name)	