System-Programmierung (syspr)

09. Juni 2022

thomas.amberg@fhnw.ch

Assessment II

Vorname:	Punkte:	_ / 90,	Note:
Name: Frei lassen für Korrektur.			ktur.
Klasse: 4ibb2			
Hilfsmittel:			
- Ein A4-Blatt handgeschriebene Zusammen	fassung.		
- Lösen Sie die Aufgaben jeweils direkt auf de	en Prüfungsblättern.		
- Zusatzblätter, falls nötig, mit Ihrem Namen	und Fragen-Nr. auf jede	em Blatt.	
Nicht erlaubt:			
- Unterlagen (Slides, Bücher,).			
- Computer (Laptop, Smartphone,).			
- Kommunikation mit anderen Personen.			
Bewertung:			
- Multiple Response: \square Ja oder \square $Nein$ anki	reuzen, +1/-1 Punkt pro	richtige/fa	alsche Antwort,
beide nicht ankreuzen ergibt +0 Punkte; To	tal pro Frage gibt es nie	weniger a	ıls 0 Punkte.
- Offene Fragen: Bewertet wird Korrektheit,	Vollständigkeit und Kürz	e der Ant	wort.
- Programme: Bewertet wird die Idee/Skizze	und Umsetzung des Pro	gramms.	
Fragen zur Prüfung:			
- Während der Prüfung werden vom Dozent	keine Fragen zur Prüfun	g beantwo	ortet.

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

Threads und Synchronisation

1) Simulieren Sie 12 Personen, die in einen *Stadtbus* mit 3 Türen, 9 Sitzplätzen und genügend Stehplätzen einsteigen. Jede Person betritt die nächste freie Türe, durchschreitet diese in 1-3s, sitzt wenn möglich sofort ab, und meldet dann "got a seat!". Das Programm soll Threads und (mehrere) Semaphoren nutzen, um Gedränge beim Einsteigen zu vermeiden. Punkte: _ / 18

```
$ ./bus

#4 got a seat!

#2 got a seat!

#3 got a seat!

#5 got a seat!

#1 got a seat!

#6 got a seat!

#7 got a seat!

#11 got a seat!

#8 got a seat!
```

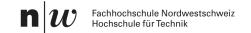
Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
int printf(const char *frmt, ...); // frmt int %d, float %f, double %lf
long random(void); // returns a value between 0 and (2^31) - 1
unsigned int sleep(unsigned int sec); // sleep for a number of seconds

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, 0 on success
pthread_t pthread_self(void); // returns the ID of the calling thread
void pthread_exit(void *retval); // terminate calling thread, no return

int sem_init(sem_t *sem, int pshared, unsigned int value); // initialize
an unnamed semaphore, pshared = 0, returns 0 on success
int sem_wait(sem_t *s); // decrement a semaphore, blocking if value <= 0,
returns 0 on success; the below non-blocking variant works the same, but
int sem_trywait(sem_t *s); // returns -1, EAGAIN instead of blocking
int sem_post(sem_t *s); // increment a semaphore, returns 0 on success</pre>
```

(Fortsetzung siehe nächste Seite)



IPC mit Pipes

2) Schreiben Sie ein Programm *len*, das die Länge des Outputs eines beliebigen Programms ausgibt, dessen Pfadname per Command Line übergeben wird. Nutzen Sie dazu eine *Pipe* und den *dup2()* Call, der *stdout* auf *fd* "umbiegt", mit *dup2(fd, STDOUT FILENO)*. Punkte: / 18

```
$ ./hello
Hello, World!
$ ./len ./hello
14
```

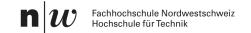
Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
int printf(const char *format, ...); // format string %s, char %c, int %d
pid_t fork(void); // create a child process, returns 0 in child process
int execve(char *pathname, char *argv[], char *envp[]); // executes the
program referred to by pathname in the calling process, passes argv, envp
pid_t wait(int *wstatus); // wait for child process to terminate
noreturn void exit(int status); // cause normal process termination

int pipe(int pipe_fd[2]); // create a pipe, from pipe_fd[1] to pipe_fd[0]
ssize_t read(int fd, void *buf, size_t count); // read from a file descr,
returns 0 (EOF) if reading a pipe which has been closed on the other end
ssize_t write(int fd, const void *buf, size_t count); // write to a file
int close(int fd); // close a file descriptor, returns 0 on success

int dup2(int old_fd, int new_fd); // the file descriptor new_fd is
adjusted so that it refers to the same open file description as old_fd
```

(Fortsetzung siehe nächste Seite)



dee (kurz) und S			

Sockets

3) Gegeben den folgenden Code, implementieren Sie in *main()* ein Programm *udp2tcp* das ein *UDP* Paket (max. 4096 Bytes) empfängt, und dessen Inhalt *per TCP* an die Absender-Adresse zurücksendet, aber auf den Port 8080, statt dem ursprünglichen Absender-Port. Falls dort ein TCP Server antwortet, soll die ganze Antwort auf die Konsole ausgegeben werden. P.: _ / 18

```
struct sockaddr_in addr;

void initAddr() {
    char *host = "0.0.0.0"; // any
    int port = 7070; // udp
    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;
}

int main(void); // TODO: implement
```

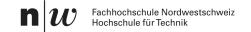
Hier ein Auszug aus der Doku, #includes und Fehlerbehandlung können Sie weglassen:

```
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 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
  int connect(int sock_fd, struct sockaddr *addr, socklen_t addrlen); //
  initiate a connection on a socket

ssize_t recvfrom(int sock_fd, void *buffer, size_t length, int flags,
  struct sockaddr *address, socklen_t *address_len); // receive a message
  from a socket; flags = 0; returns the length of the message in bytes
  ssize_t sendto(int sock_fd, void *message, size_t length, int flags,
  struct sockaddr *dest_addr, socklen_t dest_len); // send a message

ssize_t read(int fd, void *buf, size_t count); // read from a file descr.
  ssize_t write(int fd, const void *buf, size_t count); // write to a file
  int close(int fd); // close a file descriptor, returns 0 on success
```

(Fortsetzung siehe nächste Seite)



POSIX IPC

4) Nennen Sie drei wesentliche Unterschiede von Stream Sockets zu Datagram Sockets.

Formulieren Sie jeweils beide Seiten des Unterschieds aus.

Stream Sockets	Datagram Sockets

5) Schreiben Sie ein Programm $mq_carousel$, welches alle Messages aus einer per Pfadname gegebenen POSIX Message Queue herausliest, auf die Konsole ausgibt, und dann wieder in die Queue schreibt. Vereinfachung: Alle Messages haben dieselbe Priorität x. Punkte: $_/$ 12

```
$ ./mq_carousel /my_mq
hello
hola
hi
```

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

```
int printf(const char *format, ...); // format string %s, char %c, int %d
ssize_t mq_receive(mqd_t mqd, char *msg, size_t len, unsigned int *prio);
// receive a message from a message queue; returns # of bytes in message
int mq_send(mqd_t mqd, char *msg, size_t len, unsigned int prio);
// send a message to a message queue; returns 0 on success
int mq_getattr(mqd_t mqd, struct mq_attr *attr); // read attributes
struct mq_attr { ... long mq_maxmsg; long mq_msgsize; long mq_curmsgs; };
// max # of messages; max message size; current number of messages on mq
```

(Fortsetzung siehe nächste Seite)

Punkte: /6

(5) Idee (kurz) und	d Source Code hier, oder auf Zusatzblatt mit Ihrem Namen und Frage-Nr.
Zeitmessung	
6) Welche der folg	enden Aussagen zu Zeitmessung treffen im allgemeinen zu? Punkte: _ / △
Zutreffendes ankr	euzen:
□ Ja □ Nein	Die Linux Epoche ist die Zeit seit dem Aufstarten.
\square Ja \square Nein	User CPU Zeit ist immer grösser als System CPU Zeit.
□ Ja □ Nein	CPU Zeit wird relativ gemessen, an mehr als einem Punkt.
□ Ja □ Nein	Reale Zeit ist die Summe von User und System CPU Zeit.

7) Angenommen, dass 7 "Hundejahre" einem "Menschenjahr" entsprechen, schreiben Sie ein Programm dogbday, das ein Geburtsdatum (nach 01.01.1970) nimmt, und den diesem Alter entsprechenden "Hundegeburtstag" ausgibt. Das Datum soll jeweils im Format "%d.%m.%Y" in Lokalzeit geparsed und ausgegeben werden, wie im Beispiel.

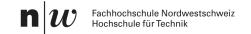
Punkte: _ / 14

```
$ ./dogbday 05.03.1975
07.09.2015
```

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

```
int printf(const char *frmt, ...); // frmt int %d, float %f, double %lf
time_t time(time_t *t); // returns the time
                                              struct tm {
as the number of seconds since the Epoch,
                                                int tm_sec; // 0-60s
1970-01-01 00:00:00 (UTC); t can be NULL
                                                int tm_min; // 0-59'
                                                int tm_hour; // 0-23h
time_t mktime(struct tm *tm); // convert
                                                int tm_mday; // Day of the
broken-down time into time since the Epoch;
                                              month 1-31
struct tm *localtime(time_t *t); // convert
                                                int tm_mon; // Month 0-11
a time value to a broken-down local time
                                                int tm_year; // Year-1900
                                                int tm_wday; // 0-6, Sun=0
char *strptime(char *s, char *format,
                                                int tm_yday; // Day in the
struct tm *tm); // convert a string repre-
                                              year 0-365, 1 Jan=0
sentation of time to a time tm structure
                                                int tm_isdst; // Daylight
size_t strftime(char *s, size_t max, char
                                              saving time
*restrict format, struct tm *tm); // format
                                              };
date and time
```

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



Zusatzblatt zu Aufgabe Nr	_ von (Name)	
		7