Fachhochschule Nordwestschweiz
Hochschule für Technik

System-Programmierung (syspr) 16. Dezember, 2025 thomas.amberg@fhnw.ch

Assessment II

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.

Threads und Synchronisation

1) Schreiben Sie ein Programm *sleepsort*, welches Zahlen sortiert, indem es für jede Zahl *n* in einem eigenen Thread *sleep(n)* aufruft, und danach die Zahl ausgibt, wie hier. Punkte: _ / 16

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

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

```
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
```

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

IPC mit Pipes

2) Schreiben Sie ein Programm, das ein per Command Line übergebenes Wort über eine Pipe vom Parent zum Child-Prozess sendet, und dort auf *STDOUT_FILENO* ausgibt. P.kte: _ / 14

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

```
void exit(int status); // cause process termination; does not return
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]
```

```
int close(int fd); // close a file descriptor
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
```

```
size_t strlen(const char *s); // calculate the length of a string
```

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

Sockets

3) Schreiben Sie ein Programm, das UNIX Domain Datagram Nachrichten unter der Adresse /echo empfängt, und den Inhalt der Nachricht an den Absender zurück schickt. Punkte: _ / 14

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

```
int bind(int socket, struct sockaddr *addr, socklen_t addrlen); // bind
ssize_t recvfrom(int socket, void *buf, size_t len, int flags, // = 0
    struct sockaddr *address, socklen_t *address_len); // receive message
ssize_t sendto(int socket, void *message, size_t len, int flags, // = 0
    struct sockaddr *dest_addr, socklen_t dest_len); // send message
int socket(int domain, int type, int protocol); // create an endpoint for
communication; domain = AF_INET, AF_UNIX; type = SOCK_STREAM, SOCK_DGRAM;
protocol = 0; returns a file descriptor for the new socket or -1 on error
```

```
struct sockaddr_un { // UNIX domain socket address structure
    sa_family_t sun_family; // AF_UNIX
    char sun_path[108]; // Pathname
};
```

```
char *strcpy(char *dest, char *src); // copy a string src to buffer dest
```

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

4) Nennen Sie drei wesentliche Unterschiede zwischen UNIX Domain Datagram Sockets und Internet Datagram Sockets (auch bekannt als UDP)? Punkte: _ / 6

Unterschiede hier eintragen, jeweils beide Seiten in einem kurzen Satz beschreiben:

UNIX Domain Datagram Sockets	Internet Datagram Sockets (UDP)

POSIX IPC

5) Welche der folgenden Aussagen über FIFOs (Named Pipes) sind korrekt? Punkte: _ / 4

Zutreffendes ankreuzen

☐ Ja | ☐ Nein Prozesse können über mehrere FIFOs Daten austauschen.

☐ Ja | ☐ Nein Bei FIFOs geht es um Message-basierten Datentransfer.

☐ Ja | ☐ Nein Ein FIFO Objekt erlaubt bidirektionalen Datenaustausch.

☐ Ja | ☐ Nein Bei *write()* blockiert ein FIFO, bis die Gegenseite bereit ist.

|

|

|

|

|

Fortsetzung auf der nächsten Seite ...

6) Schreiben Sie ein Programm, das 7 Personen simuliert, die gleichzeitig versuchen, in ein Taxi mit *3* freien Plätzen einzusteigen. Nutzen Sie Threads und Semaphoren so, dass am Ende exakt *3* Fahrgäste im Taxi sind. Diese geben "got in!" aus, alle anderen "too late". P.kte: _ / 14

Verwenden Sie die folgenden Calls, ohne #includes, mit Fehlerbehandlung, soweit wie nötig:

```
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
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
int sem_wait(sem_t *s); // decrement a semaphore, blocking if <= 0
int sem_trywait(sem_t *sem); // returns 0 on success, -1 if sem is locked
int sem_post(sem_t *s); // increment a semaphore
```

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

Zeitmessung

7) Schreiben Sie ein Programm *tomorrow*, welches "morgen", 00:00:00 als Datum (gemäss Kalender) ausgibt, im Standardformat des Betriebssystems, wie hier im Beispiel. P.kte: _ / 12

```
$ date
Tue Jun  4 13:37:11 2024
$ ./tomorrow
Wed Jun  5 00:00:00 2024
```

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

```
int printf(const char *frmt, …); // frmt int %d, double %lf, string %s
```

```
char *ctime(const time_t *t); // convert t to ASCII, default date format
struct tm *localtime(const time_t *t); // get broken-down local time
time_t mktime(struct tm *tm); // convert broken-down local time to time_t
// ignores tm_wday, tm_yday; values outside valid interval are normalised
time_t time(time_t *t); // get local time in seconds since Epoch
```

```
struct tm {
  int tm_sec, tm_min, tm_hour; // (0-60), (0-59), (0-23)
  int tm_mday; // Day of the month (1-31)
  int tm_mon;  // Month (0-11)
  int tm_year; // Year - 1900
  int tm_wday; // Day of the week (0-6, Sunday = 0)
  int tm_yday; // Day in the year (0-365, 1 Jan = 0)
};
```

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

8) Gegeben die folgenden Zeitmessungen mit *time* für die Programme, *a, b* und *c*, was sind die wesentlichen Unterschiede von *a, b* und *c,* und wie äussert sich das im Resultat? Punkte: _ / 8

```
$ cat a.c

int main() {
    sleep(3);
}



$ time ./a

real     0m3.013s
user     0m0.002s
sys      0m0.011s
```

```
$ cat b.c

int main() {
    int n =
      320000000;
    while (n > 0) {
        n--;
    }
}

$ time ./b

real     0m2.964s
user     0m2.909s
sys      0m0.020s
```

```
$ cat c.c

int main() {
    time_t t0 = time(NULL);
    time_t t1 = t0;
    while (t1 - t0 < 3) {
        t1 = time(NULL);
    }
}

$ time ./c

real     0m3.008s
user     0m0.886s
sys      0m2.082s
```

Hier ein Auszug aus der Doku:

```
unsigned int sleep(unsigned int seconds); // suspends the execution of
the calling thread; returns remaining seconds if interrupted by signal
```

```
time_t time(time_t *tloc); // get time in seconds since the Epoch
```

Unterschiede und Begründung hier eintragen; Annahme: #includes sind vorhanden.

Programm / Resultat (a)	Programm / Resultat (b)	Programm / Resultat (c)

Terminals

9) Welche der folgenden Aussagen zu Pseudoterminals treffen im Allgemeinen zu? P.kte: _ /4

Zutreffendes ankreuzen:

☐ Ja | ☐ Nein Verbindet Keyboard-Hardware direkt mit User Space Prozess.

☐ Ja | ☐ Nein Besteht aus einem Master und mehreren Subordinate Devices.

☐ Ja | ☐ Nein Erlaubt einem Prozess im User Space, Zeichen "einzutippen".

☐ Ja | ☐ Nein Ermöglicht Nutzung Terminal-orientierter Programme via SSH.

|

|

|

|

|

|

|

|

|

|

|

|

|

|

|

Zusatzblatt auf der nächsten Seite …

Zusatzblatt zu Aufgabe Nr. ____ von (Name) _____