

1. Wie kann der user mode den system calls Parameter eingeben, wenn er gar kein Zugang zu der Funktion hat?  
→ Die werden in kernel stack geladen.
2. Steht in der Trap table nur Pointer auf Systemfunktionen?  
→ Ja
3. Beim Context Wechsel wird floating point unit nicht gesichert, da der Kernel floating point frei läuft.
4. Man kann str c Signale and laufende Prozesse überschreiben.

## Fragen

### Fragen

## Frage 1

■ Die Response Time sei definiert als der Zeitpunkt, an dem die 1. Instruktion einer Task ausgeführt wird. Folgendes Scheduling 3er CPU intensiver Tasks (chapter 4 ... -1 2:100,2:100,2:100) ist gegeben:

PID: 0	PID: 1	PID: 2	CPU	I/Os
RUN: cpu	READY	READY	1	
RUN: cpu	READY	READY	1	
DONE	RUN: cpu	READY	1	
DONE	RUN: cpu	READY	1	
DONE	DONE	RUN: cpu	1	
DONE	DONE	RUN: cpu	1	

Die mittlere Response Time in diesem System ergibt sich zu:

**A: 0**  
**B: 1**  
**C: 2**  
**D: 4**  
**E: 6**

Professor Dr. Michael Mächtel

3

→  $(0 + 2 + 3) / 3 = 2$  also C

## Frage 2

- Bewerten Sie folgende Aussage:  
Die C Systembibliothek ist Teil des Betriebssystemkernels.

**Richtig**

**Falsch**

Polling options

→ Falsch

## Frage 3

- Welche Aussage ist richtig?

**A: fork erzeugt einen Kindprozess, der den Programmcode ab Zeile 1 des Programms ausführt.**

**B: fork erzeugt einen Kindprozess, der eine exakte Kopie des Elternprozesses ist.**

**C: Der fork Systemaufruf darf nicht mit dem Fehlercode -1 returnieren, da sonst die Systemstabilität gefährdet ist.**

→ B, C auf jeden Fall falsch, es gefährdet nicht die Systemstabilität. Die Benutzer Stabilität vielleicht schon.

Andere Lösung wie man richtig die Zeit berechnen kann.

```
struct timespec * diff_time( struct timespec before, struct timespec after,  
struct timespec *result )
```

```

{
    if (result==NULL)
        return NULL;
    if ((after.tv_sec<before.tv_sec) || ((after.tv_sec==before.tv_sec) &&
        (after.tv_nsec<=before.tv_nsec))) { /* after before before */
        result.tv_sec = result.tv_nsec = 0;
    }
    result->tv_sec = after.tv_sec - before.tv_sec;
    result->tv_nsec= after.tv_nsec- before.tv_nsec;
    if (result->tv_nsec<0) {
        result->tv_sec--;
        /* result->tv_nsec is negative, therefore we use "+" */
        result->tv_nsec = NANoseconds_PER_SECOND+result->tv_nsec;
    }
    return result;
}

```

Fragen die ich noch hätte:

- Warum ist context switch kleiner, als der systemcall?
- Warum wenn man println verwendet, wird dass nur zuerst in Parent ausgeführt, und dann im child. Mit sleep, wird das dann abwechselnd gemacht.