

## Betriebssysteme Übung 1+2

### Aufgabe 1

- a) fork(): Erstellt eine exakte Kopie (Sohnprozess) des aufrufenden Prozesses (Vaterprozess) und somit einen neuen Prozess

wait(): Hält den Vaterprozess an, bis Sohnprozess beendet ist

sleep(): lässt Prozess für eine bestimmte Zeit (sek.) schlafen

- b)
- 10s Sleep(Vaterprozess)
  - 30s Sleep(Vaterprozess),Sleep(1.Sohnprozess)
  - 50s wait(Vaterprozess),wait(1.Sohnprozess),sleep(1.Enkelprozess),  
sleep(2.Sohnprozess)
  - 70s wait(Vaterprozess),wait(1.Sohnprozess),sleep(1.Enkelprozess),  
sleep(1.Urenkelprozess),sleep(2.Sohnprozess),sleep(2.Enkelprozess)
  - 90s sleep(Vaterprozess),sleep(1.Sohnprozess)
  - 110s sleep(Vaterprozess),sleep(1.Sohnprozess),sleep(3.Sohnprozess),  
sleep(4.Sohnprozess)

- c) void main(void)
- ```
{  
int Pr;  
Pr = 0;  
sleep(20);  
if (Pr > 0) {printf("Vater");} else {printf("Sohn");}  
Pr = fork();  
if (Pr > 0) {printf("Vater");} else {printf("Sohn");}  
sleep(20);  
Pr = fork();  
if (Pr > 0) {printf("Vater");} else {printf("Sohn");}  
wait();  
if (Pr > 0) {printf("Vater");} else {printf("Sohn");}  
sleep(20);  
Pr = fork();  
if (Pr > 0) {printf("Vater");} else {printf("Sohn");}  
sleep(20);  
}
```

## Aufgabe 2

```
if test $# -eq 3                # wurden 3 Parameter übergeben ?
then
cd $1                          # in <StartDir> wechseln
for i in `/usr/bin/ls`         # alle Files im aktuellen Verzeichnis
do                              # in der Variable $i durchlaufen
if test -d $i                  # ist File $i ein Directory ?
then
search "$i" "$2" "$3"         # Ja → die Unterverzeichnisse
else                           # werden rekursiv durchlaufen
case "$i" in                  # Entspricht $i dem <DateiFilter>?
$2) grep "$3" $i >/dev/null && echo "$PWD/$i"
esac                          # ... falls <SuchString> ($3) im
fi                             # File $i vorkommt (-> grep gibt true zurück)
done                          # Filename + Pfad ($PWD) ausgeben!
else
echo "Benutzung: search <StartDir> <DateiFilter> <SuchString>"
fi                             # keine 3 Parameter: Hilfstext !
```

## Aufgabe 3

- Shells stellen nur die Schnittstelle zwischen Nutzer und Betriebssystem dar
- Sie nehmen Eingaben entgegen, führen Dialog mit Anwender
- Shells interpretieren lediglich die Eingabe des Nutzers für das Betriebssystem. Diese werden dann in eine Folge von Systemaufrufen übersetzt, und an das Betriebssystem übergeben.
- Die Shell kann kein Teil des Betriebssystems sein, da sie eben diese Aufgabe hat, eingegebene Kommandos in die Sprache des Betriebssystems zu übersetzen
- Die Shell ist austauschbar
- Die Shell ist Benutzerprogrammen gleichgestellt
- Entscheidet: selbst ausführen / Programm starten
- Analysiert Eingaben (zb. Masken, Variable expandieren)