Nr. 1

fork	Aufrufender Prozess (parent) dubliziert Prozess, macht also neuen Prozess auf (Kind-Prozess (child))
Execl	Ersetzt laufenden Prozess mit neuem Prozess Argument ist file, das ausgeführt werden soll
Waitpid	Unterbricht Ausführung des Prozesses bis Kind-Prozess seinen Zustand wechselt. Übergabe = ID des Kindes
Clone	Kreiert einen neuen Prozess, selber Effekt wie fork Erlaubt dem Kind-Prozess, Teile seines Ausführungskontexts mit dem Parent zu teilen (Memory, table of file, descriptors, table of signal handlers…)
System	C-Befehl, mit dem man shell-Kommandos außerhalb des eigenen Programms ausführen kann.

Nr. 2

a)	Shell Hintergrundprozess 2x Programm starten. Ein Programm STOP, danach CONT. Bei STOP wird es angehalten, bei CONT (continue) läuft es wieder weiter. Status während Prozesse schlafen: siehe Bilder.
b)	Kill-signal TERM (terminate) Prozess wird beendet.
c)	Prozess, der uns nicht gehört kill -9 Bash (als Nummer angegeben) wird geschlossen
d)	Vater-Kind-Prozesse unsortiert

Nr. 3

** Blockiert -> Laufend*

Dieser Übergang existiert nicht, weil nur das Betriebssystem entscheidet, welcher bereite Thread laufen darf, nicht der Prozess selbst.

** Bereit -> Blockiert*

Dieser Übergang existiert nicht, weil der Prozess nicht weiß, ob er blockiert ist, solange er nicht läuft.

Nr.4

pthread_create = erzeugt und startet neuen Thread
in Python: Threading importieren, Funktion machen, Thread aufrufen und Namen geben mit Argumenten und Funktion, Thread starten

pthread_join = wartet auf Ende von bestimmten Thread
in Python: join

Nr.5

BEOBACHTUNGEN:

Die globale Variable ist in allen Versuchen 0. Dies ist überraschend, weil die globale Variable nicht threadsicher ist.

Wahrscheinlich ist die Arbeit in der Thread-Schleife derart trivial, dass sie bereits abgeschlossen ist, bevor der nächste Thread die Gelegenheit hat, zu starten

Sicherlich wären auf langsameren Systemen deutliche Unterschiede bei verschiedenen Testläufen zu beobachten, bei denen alle Werte zwischen -5000 und 5000 verteilt sind.

Dies lässt sich simulieren, indem die Anzahl der Wiederholungen in der Thread-Schleife erhöht werden. Ein Wert von 1000000 war in unseren Testläufen dafür auf unserem System ausreichend.