1)

fork: Diese Funktion erstellt ein neuen Prozess, und gibt -1 bei einem Fehler. Ansonsten im Child Process wird eine 0 und im Parent Prozess die PID des childs.

excel: Diese Funktion führt ein .exe Programm aus in dem aktuellen Prozess.Kein Return Value.

waitpid: Blockiert den aktuellen Prozess bis der Prozess mit der gegebenen PID gestorben ist.

clone: wie fork, Jedoch mit geteilten Ressourcen

system: Erstellen eines Prozesses mit automatischer Behandlung, welches die manuelle Ausführung von fork/excel und deren Sonderfälle für den Nutzer übernimmt

2)

2.1)

Alle 10 Sekunden Ausgabe, dauerhaft "Running". Mit Befehl kill -stop PID wird der ausgewählte Prozess gestoppt, aber nicht terminiert.
Bei einem Stop wird die Restdauer angehalten. Bei dem Continue Befehl wird dann erst der Sleep abgearbeitet.

Während der Prozess läuft hat der Prozess trotz sleep Befehl im Code "Running". Wird er aber gestoppt wechselt er in "Stopped". Das sleep() im Pythoncode hat keinen einfluss auf die Darstellung bei dem Befehl ps.

Process verändern:

terminieren	kill PID
stoppen	kill -stop PID
starten	kill -cont PID

2.2) Man bekommt keine Ausgaben von dem Programm mehr und der Status wechselt auf terminiert. Wird bei mehrfachen "jobs"-Aufrufen wird der Prozess nicht mehr angezeigt (Existiert nicht mehr-> Keine Möglichkeit "cont" wieder zu starten).

2.3)

Antwort der Shell: "Operation not permitted".

2.4)

Prozesse werden parallel ausgeführt.

3)

blockiert -> laufend: Der Zustand blockiert wird beendet sobald die IO—operation beendet ist. Da zu diesem Zeitpunkt nicht gewährleisted werden kann, dass die CPU frei ist muss der Thread zuerst in den Zustand ready.

bereit -> blockiert: Um in den Zustand blockiert zu kommen muss das Programm einen IO-Aufruf ausführen und dies ist nur im Zustand running möglich.

4)

pthread_create() und pthread_join() werden in Python über die Klasse threading.Thread gemanagt. Dabei entspricht die Methode start() pthread_create() und die Methode join() pthread_join().

5)

Counter wird nicht gleichmäßig gezählt. Er endet nicht bei 0.

Durch das Multithreading wird der Counter parallel Hochgezählt. Dies liegt daran, dass x++ nicht atomar ist.