Aufgabe 3.2)
b)
Damit ein sogenanntes EOF (End of File) Signal den lesenden Prozess erreicht, muss der schreibende Prozess das Reading-End der Pipe schließen, sonst wird das EOF-Signal nicht gesendet. Schließt der schreibende Prozess sein Ende nicht, so wartet der lesende Prozess ständig auf das EOF. Der lesende Prozess hängt sich dann also auf.

Das Reading-End der Pipe wird vom schreiben Prozess geschlossen, weil sonst der lesende Prozess bei einem Absturz des schreibenden Prozesses kein SIGPIPE Error empfängt. Somit weiß der lesende Prozess nicht, dass der schreibende Prozess abgestürzt ist.

Aufgabe 3.3)
a)
Shared Memory
+ Es können mehr als zwei auf den geteilten Speicher zugreifen.
+ Schnelle Übertragung von großen Datenmengen

- Es muss abgesichert werden, dass Daten einzig und allein für den Empfängerprozess bereitgestellt werden.
- Synchronisation nötig

Pipes
+ Pipes sind synchron. Empfänger wartet auf Sender
+ Informationen werden immer vom gewollten Prozess gelesen oder verändert

- Kernel muss Datenströme zwischenspeichern
- Mehrere Kontextwechsel sind nötig, daher langsamere Kommunikation
- Pipes können den Leseprozess blockieren
- Gelesenes wird gelöscht. Ungeeignet, wenn mehrere Prozess die selben Daten erhalten sollen

b)
Eine Datei arbeitet nicht nach dem FIFO-Prinzip. Bei der named Pipe wird beim Zugriff das Gelesene gelöscht. Bei einer Datei ändert das Lesen die Datei in der Regel nicht.

Aufgabe 3.4)
a)
- Threads liegen innerhalb eines Prozesses.
- Threads: Mehrere unabhängige Kontrollflüsse innerhalb eines Prozesseses möglich
- Schnellerer Kontextwechsel zwischen Threads.
- Jeder Prozess hat seine eigene CPU-Zeit, während sich Threads die CPU-Zeit ihres Prozesses teilen müssen.
- Jeder Thread hat zwar seinen eigenen Stack, jedoch liegen die Stacks letzendlich in einem gemeinsamen Speicher (des Prozesses), sodass ein Thread durchaus auf den Stack eines anderen Threads zugreifen kann. Prozesse können jedoch nicht auf die Prozesse anderer Prozesse zugreifen.
- Threads können effizienter mit Threads ihres Prozesses kommunizieren, während Kommunikation zwischen zwei Prozessen langsamer ist.

b)
Da sich Threads die CPU-Zeit bzw. Speicher des Prozesses teilen, werden sie ab einer bestimmten Anzahl ineffizient.

c)
Eine Forkbomb ist ein Programm, welches rekursiv in einer Dauerschleife sich forked. Moderne Unix-Betriebssysteme begrenzen jedoch die Anzahl der Prozesse, die ein Benutzer erstellen kann und dämmt somit die exponentielle Erzeugung neuer Prozesse ein.
Außerdem verwenden sie copy-on-write, wenn also zu erzeugende Prozesse identische Informationen enthalten sollen, so wird keine neue Kopie angeleget, sondern es wird ein Pointer auf den bestehenden Prozess gesetzt.