Parallelisierung

Wir haben dafür gesorgt, dass für jede Datei, die gesucht werden soll, ein eigener Prozess gestartet wird. Das machen wir mit `fork()`. So kann jede Datei unabhängig von den anderen gesucht werden. Wenn man das Programm mit der Option `-R` startet, sucht es auch in allen Unterordnern. Durch die parallele Suche nutzen wir die Rechnerleistung besser aus, weil mehrere Prozesse gleichzeitig laufen können. Das ist besonders praktisch, wenn man viele Dateien oder große Ordnerstrukturen hat.

Synchronisation der Ausgabe

Ein Problem bei parallelen Prozessen ist, dass die Ausgaben auf der Konsole durcheinander geraten können. Wenn alle Prozesse gleichzeitig etwas ausgeben, kann das ziemlich chaotisch werden. Um das zu verhindern, haben wir dafür gesorgt, dass jeder Prozess seine komplette Ausgabe in einem Stück schreibt. Wir benutzen dafür `write()`. So kommen die Ausgaben nacheinander und nicht vermischt. Wir mussten dafür keine komplizierten Sachen wie Mutexe oder so verwenden.

Warten auf die Kindprozesse

Der Hauptprozess wartet am Ende darauf, dass alle gestarteten Prozesse fertig sind. Das machen wir mit `waitpid()`. So verhindern wir, dass sogenannte Zombie-Prozesse entstehen, die Ressourcen blockieren könnten.