Wege aus der DOSBox

Henrik Kretzschmar

henrik@nachtwindheim.de

Dieser Beitrag beschreibt eine Erweiterung für den Emulator DOSBox, mit welcher Programmteile von DOS-Programmen durch eigenen Code ersetzt werden können. Anwendungsmöglichkeiten sind die Behebung von Fehlern, kleineren Anpassungen bis hin zur kompletten Rekonstruktion des zu emulierenden Programms.

1 DOS-Programme

MS-DOS, die bekannteste DOS-Variante, wurde für den i8086 Prozessor von Intel entwickelt. Dieser Prozessor, mit einer Wortbreite von 16bit und seinem 1MB Adressraum, war allerdings ein sehr kleiner Nenner, dessen Grenzen später mittels verschiedener Brückentechnologien (EMS, XMS) mit erheblichem Mehraufwand umgangen werden mussten.

Die Tatsache, das Anwendungsprogramme die Alleinherrschaft über den PC für sich beanspruchten, fiel nicht ins Gewicht, da DOS ursprünglich als „Single Task“ System entwickelt und oft nur als solches genutzt wurde. Somit warteten die Anwendungsprogramme z.B. aktiv auf Tastatureingaben, und verschwendeten somit Prozessorzeit, die außer ihnen sowieso niemand nutzte.

Nichtsdestotrotz werden auch heute noch DOS-Programme verwendet, sei es nativ in produktiv genutzten Kassenautomaten oder Computerkassen oder von Retro-Spielern. Letztere nutzen oft den Emulator DOSBox um die alte Software auf aktuellen Systemen zu betreiben.

2 DOSBox

Das Programm DOSBox ist ein Software Emulator, welcher Intel-Prozessoren bis einschließlich des „Pentium“, einen Großteil der damals üblichen Hardware und Technologien, sowie ein DOS inklusive Kommandointerpreter emuliert. Er ist portabel, mobil und für alle gängigen Betriebssysteme verfügbar. Die verwendeten Programmiersprachen sind C und C++. Als Softwarelizenz kommt die GPL zum Einsatz. Die Erstveröffentlichung des Quellcodes war am 27.07.2002 und auch jetzt 2011 wird das Projekt aktiv weiterentwickelt.

Leider ist dieser Emulator etwas langsam, was durch den in der DOS-Ära praktizierten Programmierstil noch verschlimmert wird. Das führt auf heutigen Systemen zu hoher Systemlast, da das aktive Warten auch emuliert werden muss. Die Entwickler von DOSBox arbeiten deshalb an allgemeinen Lösungen, von denen möglichst viele emulierte Programme profitieren. Wird jedoch ein DOS-Spiel in Kombination mit DOSBox als Produkt vertrieben, kann eine spezielle Anpassung des Emulators an das Spiel eine sinnvolle Alternative sein.

3 DOSBox custom

„DOSBox custom“ ist eine Erweiterung für DOSBox, die das Überspringen von Funktionsaufrufen ermöglicht. Mit ihr ist es möglich, Code, dessen Funktionsweise bekannt ist, auf dem Host auszuführen und die Emulation zu überspringen, was bei oft genutzten und aufwendigen Funktionen einen enormen Geschwindigkeitsvorteil bringen kann.

Getestet und angewandt wurde diese Erweiterung bis jetzt bei „Real Mode“ Programmen, welche mit dem Borland C++ 3.0 Compiler übersetzt wurden. Es ist davon auszugehen, dass es auch mit Kompilaten anderer Compiler, anderen Sprachen und im „Protected Mode“ funktioniert.

„DOSBox custom“ ist als Patch für DOSBox V0.74 verfügbar. Es kann beim Kompilieren abgeschaltet werden, was in einer unveränderten DOSBox resultiert. Wird es beim Kompilieren angeschaltet wurde, hat es einen sehr geringen Overhead beim Starten und Beenden von Programmen, sowie bei “call“ Befehlen.

3.1 Initialisierung

Die Nutzung dieser Erweiterung ist nur sinnvoll, wenn das entsprechende Programm gestartet wurde. Ein neuer Prozess wird

unter DOS mit dem execute Systemaufruf erzeugt. Vom Programmierer muss eine Funktion bereitgestellt werden, die Überprüft, ob es sich bei dem soeben gestarteten Programm wirklich jenes handelt, für welches er seine Ersatzfunktionen geschrieben hat, da es sonst zu Fehlfunktionen kommen wird. Als einfache, aber sehr unsichere Variante sei die Prüfung des Dateinamens genannt. Besser ist es, zusätzlich noch nach bestimmten Zeichenketten, z.B. Versionsnummern, innerhalb des Programms zu suchen. Dem Sicherheitsempfinden des Programmierers sollen keine Grenzen gesetzt sein. Ist die Gültigkeit festgestellt und an „DOSBox custom“ rückgemeldet worden, wird die Erweiterung aktiviert.

3.2 Laufzeit

Nachdem „DOSBox custom“ zu Laufzeit aktiviert wurde, wird bei jedem „call“-Befehl eine Funktion aufgerufen, welche vom Programmierer bereitgestellt werden muss. Da es zwei unterschiedliche Arten von „call“-Befehlen gibt, die „nearcalls“, innerhalb eines maximal 64kb großen Speichersegments, und die „farcalls“, welcher sich über den gesamten Adressraum erstreckt, müssen vom Programmierer auch zwei Funktionen bereitgestellt werden. Diese beiden Funktionen werden als „nearcall“- und „farcall“-Handler bezeichnet. Die Aufgabe beider Funktionen ist es zu Überprüfen ob es sich bei der aufzurufenden emulierten Funktion um jene handelt, welche vom Programmierer gewünscht ist. Dies geschieht durch Vergleichen der Segment und Offsetadressen.

Eventuell nutzt das Programm auch die Overlay-Technik von Borland, mit welcher Programmcode-Segmente bei deren Aufruf dynamisch nachgeladen werden können. Das hat jedoch zu Folge, dass sich die Segmentadressen ändern können, was eine Prüfung bei „nearcalls“ erschwert. Als Brücke zwischen dem statischen und dem dynamischen Code dient ein kleines Codesegment mit fester Segmentadresse, welches „Stub“ genannt wird. Es dient als Einsprungspunkt für „farcalls“, hinter dem sich ein Sprungbefehl zu dem wirklichen Position befindet, falls das Overlaysegment schon im Speicher ist. Ist es nicht im Speicher, befindet sich an der Stelle ein Aufruf an den Interrupt 0x3f, hinter dem sich der Overlaymanager von Borland verbirgt. Dieser schafft, falls nötig, freien Platz, läd das angeforderte Codesegment nach, setzt die Adressen der Sprunganweisungen im Stub und führt anschließend die gewünschte Funktion aus. Da sich der „Stub“ immer an einer festen Position befindet, kann über diesen die momentane Adresse des Segments ermittelt werden. Stimmt diese mit der aktuellen Codesegmentadresse im Register CS überein, ist die gesuchte Funktion gefunden. Zum ermitteln dieser Adresse gibt es in „DOSBox custom“ ebenfalls ein Funktion, die dem Programmierer das Leben erleichtern kann.

Parameterübergabe

Die Aufrufparameter der gewünschten Funktion befinden sich auf dem Stack der DOSBox-CPU. Hält sich das DOS-Programm an die C-Konvention, so befindet sich der erste, links stehende Parameter als erstes auf dem Stack und kann mit dem DOSBox Befehl CPU_Pop16() oder CPU_Pop32() von der DOSBox-CPU ermittelt werden und sollte in lokalen Variablen gespeichert werden. Diese Befehle passen auch den Stackzeiger (Register SP) der DOSBox-CPU entsprechend an.

Laut C-Konvention räumt der Aufrufer den Stack wieder auf, weswegen der Programmierer die ursprüngliche Position des Stackzeigers wiederherstellen muss.

Eine Möglichkeit ist mit den Befehlen CPU_Push16() und CPU_Push32() in umgekehrter Reihenfolge die Werte der lokalen Variablen wieder auf den Stack zu schreiben. Es spricht auch nichts dagegen die Position des Stackzeigers vor den CPU_Pop() Befehlen zu speichern und danach wieder ins SP Register der DOSBox CPU zu schreiben, da die Daten auf dem Stack nicht beschädigt werden.

Die Firma Borland hat allerdings einen Trick genutzt, welcher bei den „nearcalls“ eine Sonderbehandlung erfordert. Das Gegenstück zu den „call“-Befehlen bilden die „ret“-Befehle, von denen es auch eine „near“ und „far“ Variante gibt. Um Funktionen sowohl innerhalb eines Segments und von Außerhalb aufrufen zu können, wurde nur der „farret“-Befehl benutzt. Wird diese Funktion von innerhalb des Segmentes aufgerufen, wird das aktuelle Codesegment auf den Stack gepusht und die Funktion mit einem „nearcall“ aufgerufen. In diesem Fall befindet sich die Adresse des aktuellen Codesegmentes auf dem Stack und muss vom Programmierer manuell entfernt werden. Wenn er die gewünschte Funktion überspringen möchte, darf er diesen Wert nicht wieder auf den Stack zurücklegen, da Dieser, entgegen den C-Konventionen, vom Aufgerufenen entfernt würde. Soll die Funktion von der DOSBox-CPU emuliert werden, muss der Adresse des Codesegments wieder auf den Stack.

Eigener Code

Nun, da man die Werte der Parameter hat, kann man die Ersatzfunktion mit diesen Werten aufrufen. Möchte der Programmierer das DOS-Programm beobachten, kann er eine entsprechende Meldung, z.B. mit printf() an der Konsole ausgeben oder in eine Logdatei schreiben.

Rückgabewert

Wurde eine Funktion, die einen Wert zurück gibt, überbrückt, muss dieser Wert noch an die DOSBox-CPU übermittelt werden. Dazu dienen für 8- und 16-bit Werte das Register AX, welches über die Variable reg_ax zu erreichen ist. Bei 32bit-Werten, wozu auch Zeiger zählen, muss der niederwertige Teil oder der Offset ins Register AX, der höherwertige Teil ins Register DX. Dieses

ist über die Variable reg_dx erreichbar.

Überspringen oder Emulieren

Als letztes muss „DOSBox custom“ informiert werden, ob der Funktionsaufruf übersprungen werden, oder DOSBox mit der Emulation der Funktion fortfahren soll. Der Rückgabewert 0 steht für Emulieren, der Rückgabewert 1 steht für das Überspringen der Funktion des DOS-Programms.

3.3 Beenden

Das Beenden von „DOSBox custom“ erfolgt über den exit() DOS-Systemaufruf, innerhalb von DOSBox. Wird dieser ausgeführt muss vom Programmierer eine Aufräumfunktion bereitgestellt werden, die den Zustand vor dem Initialisieren wieder herstellt.

4 Zusammenfassung

Die Lösung „DOSBox custom“ ist sehr speziell, da sie Kenntnisse der Programmiersprache C, der Funktionsweise von Intelprozessoren der x86 Reihe, sowie eine eingehende, statische Untersuchung des DOS-Programms voraussetzt. Die ersten beiden Fähigkeiten können universal vorausgesetzt und anderweitig genutzt werden. Das statische Untersuchen muss für jedes einzelne Programm durchgeführt werden. Gibt es verschieden Versionen dieser Programme welche unterstützt werden sollen, muss jede Version untersucht und für jede Version je ein „near“- und ein „farcall“-Handler bereitgestellt werden.

Als Einschränkung sind zu nennen, dass die Einsprungspunkte der aufzurufenden Funktionen konstant sein müssen. Es darf sich somit nicht um selbst-modifizierende Programme handeln.

Als weiterer Nachteil kann sich herausstellen, dass die ersetzten Funktionen keine zu emulierenden Funktionen aufrufen dürfen. Softwareinterrupts, sowie andere ersetzte Funktionen sind erlaubt. Somit kann es sein, dass die zu ersetzende Funktion des DOS-Programms viele andere Funktionen aufruft, welche ebenfalls erst ersetzt werden müssen, bevor man die eigentliche Funktionen ersetzen kann. Dieser Aufwand muss im Vorfeld bei der statischen Analyse des Programms ermittelt werden.

Die Lösung eignet sich am Besten für eine komplette, schrittweise Rückentwicklung von DOS-Software. Der große Vorteil daran ist, dass während des gesamten Rückentwicklungsprozesses immer eine funktionierende Version vorliegt, da nach jedem ersetzten Programmteil die Funktionalität überprüft werden kann.

Literatur

- [1] Website von DOSBox

  http://www.dosbox.com