Dokumentation zu Diffy und Patchy

Christian Wempe

COLLABORATORS			
	TITLE : Dokumentation zu Diffy und Patchy		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Christian Wempe	March 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Dokumentation zu Diffy und Patchy

1.1 Dokumentation zu Diffy und Patchy Version 2.0

1.2 einführung

Diffy ist ein kleines Programm, das die Unterschiede zwischen zwei Dateien herausfindet und diese in einem Archiv abspeichert. Mit Hilfe der Patchfunktion von Diffy, oder mit dem Programm Patchy kann dann die eine Datei gepatcht werden, so daß daraus die andere Datei entsteht. Der häufigste Anwendungsfall hierfür ist wohl, ein Patchfile zu erzeugen, daß aus einer alten Version eines Programms die neue erstellt. Rein theoretisch ist es aber natürlich möglich, hierfür zwei beliebige Dateien zu nehmen.

Das Ziel bei der Entwicklung von Diffy war hauptsächlich, die erzeugten Patchfiles so klein wie möglich zu halten, weniger die Ausführungsgeschwindigkeit. Mit anderen Worten, es gilt, so viel wie möglich von der einen Datei in der anderen zu finden, und die so gewonnene Information möglichst knapp zu codieren. Da Informationen der einen Datei in der anderen ganz wo anders stehen können (permutierte Blöcke), heißt das genau genommen, daß für jede Bytefolge überprüft werden muß, ob sie in der anderen Datei irgendwo vorhanden ist. Zusätzlich kann natürlich nicht gleich die erste genommen werden, die gefunden wird, sondern die längste. Mit

dem naheliegenden Algorithmus, der alles aus der Ursprungsdatei, von vorne angefangen, in der Zieldatei sucht, würde dies viel zu lange dauern. Daher haben wir uns einen Algoritmus ausgedacht, der vor allem eines kann: Beliebige große Blöcke aus der einen Datei schnell in der anderen finden. Sollte es keine solchen größeren Blöcke geben, kann Diffy immer noch sehr lange brauchen. Die Laufzeit hängt also sehr von der Ähnlichkeit der Datein ab.
Außerdem ist es durch den Suchvorgang notwendig, beide Dateien die ganze Zeit im Speicher zu halten.

Von Diffy wird ein Archiv erzeugt, in dem die Patchdaten, von einem oder mehreren Dateipaaren stehen. Zuerst hatten wir eigentlich an ein Pogramm gedacht, daß für jedes Dateipaar eine Patchdatei erzeugt. Dann haben wir aber festgestellt, daß man bei der Entwicklung eines Programmpaketes dann unter Umständen ziemlich viele Patchdateien weitergeben muß und vor allem, daß es auch zu Namenskollisionen kommen kann. Kurz, es ist halt einfacher, ein Archiv weiterzugeben, daß die Patchdaten für alle Dateien enthält, die sich geändert haben.
Inzwischen, seit der Version 2.0, kann Diffy auch ganze Verzeichnisbäume, inklusive neuer und alter Unterverzeichnisse und Dateien erzeugen.

Noch ein Wort zu den von mir verwendeten Bezeichnungen: Wenn ich in dieser Dokumentation von einer alten Datei und einer neuen Datei spreche, meine ich damit, daß die neue Datei diejenige ist, die mit Hilfe des Patchfiles, das Diffy aus diesen beiden erzeugt, aus der alten Datei erzeugt werden kann. Sie muß nicht unbedingt wirklich 'älter' sein, es wären ja auch Patches denkbar, bei denen ältere Versionen aus der aktuellen erzeugt werden.

1.3 aufruf von diffy

Mit Diffy lassen sich fünf verschiedene Aktionen ausführen, die teilweise mit Optionen beeinflußt werden können:

Patches zu einem Archiv hinzufügen

Patches aus einem Archiv löschen

Archivinhalte auflisten

Patches aus einem Archiv anwenden

Testen, wie sich ein Patch auswirkt

Optionen: -b<n> -o<name> -q -r -x
-c -e -h -n -z<path>

1.4 aktion add

Die Aktion ADD dient dem Hinzufügen von Patches in ein Archiv. Hierfür gibt es zwei verschiedene Arten des Aufrufs.

1. Diffy [-a] [-oArcName] [-bN] [-q] [-c] [-h] file.alt file.neu

Ein Patch von file.alt nach file.neu wird in das Archiv aufgenommen. Der Archivname ist ArcName, falls die Option -o angegeben ist, ansonsten file.dfy.

2. Diffy [-a] [-oArcName] [-bN] [-q] [-r] [-x] [-c] [-e] [-h] [-n]
dir.alt [(file|dir).neu1 (file|dir).neu2 ...]

Hier wird 'dir.alt' paarweise mit den nachfolgenden Parametern verarbeitet. Ist einer dieser Parameter eine Datei, so wird sie in dir.alt (direkt und ohne Pfad) gesucht und diese beiden dann verglichen. Falls sie dort nicht vorhanden ist, so wird sie ignoriert, oder (bei Option -n) ganz aufgenommen.
Ist einer der weiteren Parameter ein Verzeichnis, denn wird die gesamten Verzeichnisbäume dir.alt und dir.neuX verglichen. Falls die Option -r nicht angegeben wurde, werden auch alle Unterverzeichnisse durchlaufen. Die Option -e wird nach dem ersten durchlaufen zweier solcher Verzeichnisse automatisch abgeschaltet, damit keine Dateien als 'zu löschend' eingetragen werden, die in dir.alt existieren und nicht in ALLEN anderen.
Es ist sowieso sinnvoll, bei einem Verzeichnis mit neuen Dateien zu blieben. Mehrere Verzeichnisse anzugeben ist dann sinnvoll, wenn man eine Menge von alten Dateien in einem Verzeichnis gesammelt hat, z.B. für eine ganz bestimmte Person. Dann sollte man sich aber auf reine Diffs beschränken und sowohl die Option -n als auch die Option -e nicht verwenden. Beispiel:
diffy -oUpdate.dfy Holger/ PacShell/ Diffy/ Hcp/

Da die Aktion ADD die Default-Aktion ist, ist die Angabe von -a optional.

1.5 aktion delete

Die Aktion DELETE dient dem Löschen von Patch-Modulen aus einem Archiv. Der Aufruf lautet:

Diffy -d [-q] [-x] -oArcName pmod1 pmod2 ... oder auch
Diffy -d [-q] [-x] ArcName pmod1 pmod2 ...

Beide Aufrufe sind äquivalent und aus dem Archiv ArcName werden die Module pmod1, pmod2 usw. entfernt. Wird die Option -x angegeben, so werden beim Suchen der Module im Archiv alle vorhandenen Pfadnamen benutzt.

Achtung: Da ein Archiv mehrere Module mit gleichem Namen enthalten kann, und man beliebig viele davon löschen können sollte, werden so viele davon gelöscht, wie oft der Name in der Kommandozeile angegeben wurde, angefangen beim vordersten.

Beispiel: Diffy -d ArcName test test

Löscht die ersten beiden Module mit Namen test.

1.6 aktion list

Die Aktion LIST dient dem Auflisten von Archivinhalten. Der Aufruf lautet:

Diffy -l [-q] [-o]ArcName1 file1 file2 ...

Listet die Dateien file1, file2 usw. aus dem Archiv ArcName1. Wird nur ein Archivname angegeben, werden alle Dateien gelistet.
Folgende Daten werden hierbei ausgegeben:

Datalen — Die Länge des Datenbereiches dieses Moduls. Das sind die Daten, die in der alten Datei nicht gefunden wurden und daher in das Patch-Modul aufgenommen wurden.

Codelen — Die Länge des Codebereiches dieses Moduls. Das sind die Strukturierungsdaten, die angeben, was woher und wohin kopiert werden muß.

Total — Die Summe aus den beiden vorherigen Längen.

CRC — Ein CRC-Wert für dieses Modul. Dies ist die Prüfsumme der alten Datei und hiermit wird vor dem Patchen überprüft, ob es sich auch um die richtige Datei handelt, die da gepatcht wird. Dieser Wert wird nur bei alten Headern ausgegeben. Da er eher uninteressant ist, habe ich ihn bei neuen Headern aus Platzgründen weggelassen.

Attrib — Die Dateiattribute des Moduls, falls vorhanden.

Date — Das (neue) Datum der Datei

Time — Die (neue) Uhrzeit der Datei

Name — Der Dateiname des Moduls.

Am Ende wird noch die Summe aller Datenbereiche und aller Codebereiche ausgegeben.

Die Option -q unterdrückt hier nur die Copyright-Meldung.

1.7 aktion patch

Die Aktion PATCH dient dazu, mit Hilfe eines Patch-Archives, Dateien (selektiv) zu patchen. Der Aufruf lautet:

Diffy -p [-q] [-x] [-z<path>] -oArcName [pmod1 pmod2 ...] oder
Diffy -p [-q] [-x] [-z<path>] ArcName [pmod1 pmod2 ...]

Patcht die Dateien pmod1, pmod2 usw. mit Hilfe der Patchdaten aus dem Archiv ArcName. Werden keine Dateinamen angegeben, so werden alle Dateien gepatcht, die gefunden werden und für die Patchdaten im Archiv vorhanden sind.
Wird die Option -x angegeben, so werden beim Vergleich der angegebenen Dateinamen und denen im Archiv, die Pfadnamen benutzt. Von dem angegebenen Dateinamen wird der Pfad natürlich nie entfernt, sondern die Datei, wie angegeben, gepatcht.

1.8 aktion test

Die Aktion TEST dient dazu, zu testen, wie sich das Patchen auswirken würde. Für jeden Eintrag im Archiv testet Diffy, ob das ausführen des Patches hier Erfolg haben dürfte.

Diffy -t [-x] [-z<path>] -oArcName [pmod1 pmod2 ...] oder
Diffy -t [-x] [-z<path>] ArcName [pmod1 pmod2 ...]

Testet, ob das Patchen der Dateien pmod1, pmod2 usw. aus dem Archiv ArcName Erfolg haben wird. Werden keine Dateinamen angegeben, so wird das gesamte Archiv getestet.
Wird die Option -x angegeben, so werden beim Vergleich der angegebenen Dateinamen und denen im Archiv, die Pfadenamen beutzt.

1.9 option -b

Mit der Option -b kann man (eingeschränkt) die Blockgröße festlegen, die Blöcke mindestens haben müssen, damit Diffy sie in der gesamten Datei sucht. Diese Suche erfolgt für alle Blöcke, die noch übrig geblieben sind, nachdem zuerst nur Blöcke gesucht wurden, deren Position in beiden Dateien die gleiche Reihenfolge haben.
Diese erweiterte Suche kann viel Zeit in Anspruch nehmen, falls noch viel übrig ist und daher kann man mit dieser Option die Geschwindigkeit auf Kosten einiger Bytes mehr in der Patchdatei beeinflussen. Mit -b0 wird diese Suche ganz ausgeschaltet und je höher der Wert (maxmial 9), desto kleiner dürfen diese Blöcke werden und desto länger dauert es dann im Allgemeinen. Der voreingestellte Wert ist natürlich 9, denn wir wollen ja möglichst kleine Patchfiles haben.

Beispiel: Diffy -b0 prog.alt prog.neu

1.10 option -o

Mit der Option -o kann der Name des Archivs festgelegt werden, in das die Patchmodule geschrieben werden. Wird keine Archivname angegeben, wird der Archivname aus der ersten behandelten neuen Datei gebildet, indem bei diesem die Endung durch "dfy" ersetzt

wird.

Beispiele:
Diffy -oarchiv.dfy prog.alt prog.neu Archivname = archiv.dfy
Diffy prog.alt prog.neu Archivname = prog.dfy

Beim Patchen (-p), Testen (-t), Listen (-l) und Löschen (-d) in Archiven ist die Angabe des Archivnamens mit -o zwar möglich, aber nicht notwendig. In dem Fall wird der erste Dateiname als der Name des Archivs angesehen.

Zwischen der Option und dem Archivnamen dürfen auch Blanks stehen, was insbesondere nützlich ist, wenn der Name Blanks enthält und man ihn in Anführungszeichen setzen muß.

1.11 option -q

Mit der Option -q werden fast alle Ausgaben von Diffy abgeschaltet, so daß dieser im Normalfall keine Ausgaben macht.
Fehlermeldungen und natürlich die Ausgaben des List-Kommandos werden natürlich immer noch ausgegeben.

Wird die Option mehr als einmal angegeben, so ist diffy wirklich völlig ruhig und macht überhaupt keine Ausgaben mehr, auch keine Fehlermeldungen oder sonstwas.
Sowas ist also völlig sinnlos: diffy -l -qq archiv.dfy

1.12 option -r

Die Option -r hat nur beim Hinzufügen von Patches in ein Archiv eine Wirkung und da auch nur, wenn es sich um Verzeichnisse handelt, die durchsucht werden. Sie verhindert, daß auch alle Unterverzeichnisse durchsucht werden.

Beispiel: Diffy new old

In diesem Fall würden in dem Verzeichnis old auch alle enthaltenen Unterverzeichnisse durchsucht werden. Existiert dort z.B. eine Datei (mit Pfad) old/prog/start, so würde nach new/prog/start gesucht werden und, falls gefunden, ein Patch zwischen den beiden in das Archiv abgelegt werden, mit Namen prog/start.
Mit 'Diffy -r new old' würde das ganze Verzeichnis prog jedoch ignoriert werden.

Moch eine Anmerkung:
Wird die Option -n angegeben und die Option -r nicht, so werden beide Verzeichnisbäume nacheinander rekursiv durchsucht. Zuerst wird immer das Verzeichnis mit den alten Dateien durchsucht und Dateien, die in beiden Verzeichnissen vorhanden sind, werden gedifft und die, die nur im alten Verzeichnis stehen werden (bei Option -e) als zu löschend eingetragen.
Um aber herauszufinden, welche Dateien neu hinzugekommen sind, muß

(falls die Option -n angegeben ist) das andere auch noch durchsucht werden.

1.13 option -x

Normalerweise werden Pfade nur in das Archiv aufgenommen, wenn auch Unterverzeichnisse durchsucht werden. Wird die Option -x angegeben, werden Pfade generell benutzt, allerdings immer ohne die Laufwerksbezeichnung. Beispiel:

```
diffy -xn drive:project/diffy/diffy.guide
```

fügt die Datei mit Namen project/diffy/diffy.guide ganz in das Archiv ein.

1.14 option -c

Normalerweise werden die alte und die neue Datei getauscht, wenn die neue Datei ein älteres Datum hat. Dies ist insbesondere dann nützlich, wenn man Diffy von einer grafischen Oberfläche aus beutzt und nicht weiß, in welcher Reihenfolge die Dateien nun wirklich übergeben werden. Mit der Option -c kann dies abgeschaltet werden.

1.15 option -e

Mit der Option -e können Dateien in das Archiv aufgenommen werden, die beim Auspacken gelöscht werden sollen.

Benutzt man diese Option, wenn Dateien direkt übergeben werden, dann werden alle Dateien als zu löschende Dateien in das Archiv eingetragen:

```
diffy -e -otest.dfy file1 file2 file3
```

Die Dateien file1, file2 und file3 werden als zu löschende Dateien in das Archiv eingetragen. Diese Dateien werden immer eingetragen und müssen nicht unbedingt wirklich vorhenden sein.

Falls zwei Verzeichnisse gescannt werden, dann werden alle Dateien, die im alten Verzeichnis existieren und im neuen nicht existieren zum Löschen eingetragen, falls -e angegeben ist.
Da diese Option keinen Sinn ergibt, wenn mehr als ein Verzeichnis mit neuen Dateien angegeben ist, wird sie nach dem Vergleich von den ersten beiden Verzeichnissen ausgeschaltet. Anderenfalls würde eine Datei aus olddir jedes mal als 'zu löschend', wenn sie in einem der Verzeichnisse nicht existiert.

```
diffy -e -otest.dfy olddir dir1 dir2
```

Hier werden nur Dateien als 'zu löschend' eingetragen, die in olddir existieren und in dir1 nicht.

1.16 option -h

In dieser Version kennt Diffy ein neues, zweites Headerformat , das er standardmäßig auch benutzt. Mit der Option -h kann man Diffy dazu bringen, das alte, kürzere Format zu benutzen.
Diese Option wird ignoriert, falls man Dateien zu einem existierenden Archiv hinzufügt, denn dann wird automatisch das Format benutzt, das in diesem Archiv benutzt wurde.
Der Vorteil des alten Formats ist, daß es etwas kürzer ist und die Archive dadurch ein wenig kleiner werden. Wenn es aufs Byte ankommt, dann kann man also das alte Format benutzen.
Das neue Format hingegen ist sicherer und bietet ein paar Möglichkeiten mehr, die man beim alten Format nicht benutzen kann. Wenn möglich, sollte man also das neue Format benutzen.
Beim Auspacken wird automatisch erkannt, welches Format das Archiv hat.

1.17 option -n

Mit der Option -n kann man ganze Dateien in ein Archiv aufnehmen. Wenn man also ein Paket difft, das aus vielen Dateien besteht und auch noch ein paar neue dazu gekommen sind, sollte man diese Option benutzen, um die neuen Dateien ganz in das Archiv aufzunehmen.

Beim Aufruf mit einzelnen Dateien, werden diese alle ganz in das Archiv aufgenommen, wenn diese Option angegeben ist:

```
diffy -n -otest.dfy file1 file2 file3
```

Hier werden file1, file2 und file3 ganz in das Archiv aufgenommen. Wird hier gleichzeitig die Option -e angegeben, so wird diese ignoriert.

Werden zwei Verzeichnisse gescannt, dann werden, falls diese Option angegeben ist, alle Dateien ganz aufgenommen, die in dem neuen Verzeichnis existieren und im alten nicht.

Gibt man ein Verzeichnis und eine Liste von neuen Dateien an, werden natürlich die Dateien ganz aufgenommen, die in dem Verzeichnis nicht existieren:

```
diffy -n -otest.dfy olddir/ file1 file2 file3
```

Angenommen, file2 existiert in olddir nicht, dann wird sie ganz aufgenommen, sonst mit der gleichnamigen dort verglichen.

1.18 option -z

Mit der Option -z kann der Pfad angegeben werden, wo die zu patchenden Dateien stehen. Diese Option hat nur eine Wirkung, wenn Diffy zum Patchen oder Testen verwendet wird.
Wird diese Option nicht angegeben, so wird das aktuelle Verzeichnis genommen.

Beispiele:

```
diffy -p -zTools:Prog1 update.dfy
Patcht die Dateien in Tools:Prog1 mit Hilfe der Patchdaten in
update.dfy

diffy -p -zTools:Prog1 update.dfy
Testet, wie sich ein Patch mit Hilfe der Patchdaten aus update.dfy
in dem Verzeichnis Tools:Prog1 auswirken würde

Zwischen der Option und dem Pfad dürfen auch Blanks stehen, was
insbesondere nützlich ist, wenn der Pfad Blanks enthält und man
ihn in Anführungszeichen setzen muß.
```

1.19 das minimalprogramm patchy

```
Dieses Programm kann nur Diffy-Archive lesen und alle darin
angegebenen Dateien patchen. Dieses Programm hat keine Optionen
und das einzigste, was beim Aufruf angegeben werden muß, ist ein
oder mehrere Archivnamen. Der Aufruf lautet also:

    Patchy <archiv1> [<archiv2> ... ]

Alle in den Archiven vorhanden Patches werden hiermit ausgeführt,
sofern die entsprechenden Dateien gefunden werden. Selektives
Patchen ist hiermit nicht möglich, dazu muß man Diffy verwenden.

Die alten Dateien werden, wie bei Diffy, vorher im Verzeichnis
"PatchBak" gerettet. Dieses Verzeichnis legt Patchy im aktuellen
Verzeichnis an, falls es noch nicht existiert. Unterverzeichnisse
werden hierin nicht angelegt. Bei Archiven mit Pfaden, in denen
zwei verschiedene Dateien mit selben Namen aber unterschiedlichen
Pfaden enthalten sind, kann also nur eine Datei gesichert werden.

Siehe auch: Patchen mit Diffy
```

1.20 wichtige hinweise

```
Folgende Dinge sollte man bei der Verwendung von Diffy unbedingt
beachten:

Diffy ist kein richtiger Archiver
Das heißt insbesondere, daß neue Module immer hinten angehängt
werden, auch wenn ein gleichnamiges schon vorhanden ist. Die
vorgesehene Anwendung ist eher, daß man ein Diffy-Archiv entweder
Dateipaar für Dateipaar, oder auf einmal erzeugt und dieses dann
weitergibt. Das ein Archiv mehrere Module enthält, hat halt nur
den Grund, daß man nicht mehrere Dateien weitergeben muß.

Bei sehr unähnlichen Dateien dauerts lange
Dieser Umstand ist durch den Algoritmus bedingt. Das wichtigste
war halt, möglichst viel zu finden, damit die Patchdateien schön
kurz werden. Da zuerst hintereinander liegende Blöcke gesucht
```

werden und erst danach entfernte Blöcke, kann man auf letzteres mit der Option -b Einfluß nehmen, zumal hierfür mehr und mehr Zeit benötigt wird, je unähnlicher die Dateien sind.

Komprimierte Dateien zu vergleichen, sollte man lassen
Besonders bei Huffman-codierten Dateien hat die kleinste Änderung schon solche Auswirkungen, daß praktisch die ganze komprimierte Datei andes aussieht, wenn man sie Byte-weise betrachtet. So gesehen sind sie sehr unähnlich für Diffy und es würde sehr sehr lange dauern den Vergleich durchzuführen und das Resultat wäre wegen seiner Länge auch nicht befriedigend. Von Diffy erzeugte Patchdateien lassen sich dagegen immer noch ganz gut packen. Also erst ein Diffy-Archiv erzeugen und dann dieses kompimieren und nicht die koprimierten Versionen zweier Dateien mit Diffy vergleichen. Seit der Version 2.0 ist Diffy aber gerade hier deutlich schneller geworden.

Mehrere Module gleichen Namens in einem Archiv
Zuerst einmal, man sollte dies lieber vermeiden, auch wenn es möglich ist. Wenn überhaupt, dann sollte man es nur verwenden, um meherer Revisionen einer Datei in einem Archiv zu halten.
Man sollte sich mal genau vor Augen führen, was das heißt und welche Möglichkeiten sich daraus ergeben. Zuerst einmal wird beim Patchen das Archiv von vorne nach hinten durchgearbeitet. Wird die Datei gefunden, wird über die CRC getestet, ob es sich auch um die richtige Version handelt. Wenn ja, wird sie gepatcht und danach existiert eine Datei gleichen Names, aber mit neuerem Inhalt. Hat man von einem Programm die Versionen Prog1, Prog2, Prog3 und Prog4, ist es einerseits möglich, in ein Diffy-Archiv Patches von Prog1 nach Prog2, von Prog2 nach Prog3 und von Prog3 nach Prog4 abzulegen. In dem Fall würde dann halt mehrfach gepatcht werden. Andereseits ist es auch möglich, Patches von Prog1 nach Prog4, von Prog2 nach Prog4 und von Prog3 nach Prog4 abzulegen. In dem Fall würde nur einmal gepatcht, es ist aber möglich, dies für mehrere alte Versionen zu ermöglichen. Wenn man dies ausnutzt, sollte man sich aber nochmal vergewissern, daß die CRC-Werte unterschiedlich sind (Option -l zum Archiv listen). Außerdem lohnt sich das nur, wenn die Summe der Patches kleiner bleibt als die neueste Version, oder man wirklich mehrere Revisionen einer Datei aufheben möchte.

1.21 rechtliches

Diffy v1.0 ist weiterhin Freeware, d.h. jeder kann es benutzen, ohne dafür bezahlen zu müssen, das Copyright bleibt aber bei uns.

In der Version 2.0 habe ich den Status auf Fairware geändert, da in dieser Version doch schon eine Menge Arbeit steckt. Das heißt also, Diffy ist zwar uneingeschränkt benutzbar, doch wer es öfter verwendet, sollte mir der Fairness halber eine kleine Spende zukommen lassen.

Die Benutzung geschieht auf eigene Gefahr. Für eventuelle Fehler im Programm und Schäden, die durch die Benutzung enstehen, können wir natürlich keine Haftung übernehmen.

Autoren: Christian Wempe
St. Antoniort 28
49610 Quakenbrück

Holger Weets
Tangastr. 45
26121 Oldenburg
(Atari-Anpassung)

E-Mail bitte an Holger: Holger Weets@OL (Mausnetz)
Holger_Weets@OL.maus.de (Internet)

Bankverbindung: Kreissparkasse Bersenbrück
BLZ: 26551540
Konto-Nr.: 0118789197
Empfänger: Christian Wempe

--Christian

1.22 neuigkeiten

Neues in Version 2.0

Diverse Optimierungen und Erweiterungen am Diff-Algorithmus. Das Suchen läuft jetzt i.A. schneller als in der alten Version. Besonders stark betroffen von der Geschwindigkeitssteigerung sind die Dateien, wo sich die alte Version sehr viel Zeit gelassen hat, also komprimierte Dateien. Allerdings werden hierfür 64kB Speicher zusätzlich benötigt und wenn die nicht mehr zur Verfügung stehen, dauert es in etwa genau so lange wie früher. Wie hoch die Geschwindigkeitssteigerung ist, läßt sich nicht genau sagen, in manchen Fällen sind es wenige Prozent und in anderen kann es durchaus mal um den Faktor 100 oder noch schneller gehen.

Der Algorithmus ist und war nicht darauf ausgelegt, das optimale Ergebnis zu liefern, sondern eines das dem möglichst nahe kommt. Durch ein paar Änderungen, kommt das Ergebnis dem Optimum nun näher und dadurch werden die Diffs meistens ein wenig kürzer.

Es gibt ein neues und erweitertes Headerformat . Dieses ist zwar ein wenig länger, doch bietet es mehr Sicherheit und erweiterte Funktionalität.

Diese Version ist darauf ausgelegt, auch ganze Verzeichnisbäume zu vergleichen. So können jetzt auch ganze Dateien aufgenommen werden, zu löschende vermerkt werden und auch Unterverzeichnisse gelöscht oder erzeugt werden.

Falls kein Archivname angegeben wird, dann wird dieser Name jetzt aus einem Verzeichnisnamen erzeugt, falls Diffy mit einem Verzeichnisnamen für alte Dateien aufgerufen wird.

Es gibt einen Testmodus (Kommandooption -t), mit dem man testen kann, ob ein Patch auch funktionieren würde. Hier wird alles gemacht, was beim Patchen auch gemacht würde, nur das keine Dateien wirklich gepatcht, gelöscht oder extrahiert werden.

Man muß beim Patchen (und Testen) jetzt nicht mehr in dem Verzeichnis sein, wo gepatcht werden soll. Dieses Verzeichnis läßt

sich jetzt auch in der Kommandozeile mit der Option -z angeben.

Es gibt nur noch zwei verschiedene Aufrufschemata, denn drei davon (olddir ; olddir newdir ; olddir new1 new2 ...) wurden zu einem zusammengefasst: olddir [new(dir)1 new(dir)2 ...]. Dadurch hat sich dir Anzahl der möglichen Aufrufschemata aber nicht verringert, sondern erweitert!

Durch mehrfache Angabe von -q kann man diffy jetzt vollkommen ruhig stellen und es werden nicht mal mehr Fehlermeldungen ausgegeben.

Die Bedeutung der Option -r wurde umgedreht: Defaultmäßig werden Verzeichnisse jetzt rekursiv (mit Unterverzeichnissen) durchsucht, was mit der Option -r verhindert werden kann.

Das Format des Listkommandos wurde geändert. Es können jetzt nicht mehr mehrere Archive auf einmal gelistet werden, sondern einzelne Dateien selektiv aus einem Archiv.

Bei Optionen mit einem Argument, wie -oArchive.dfy, darf jetzt auch ein Blank zwischen der Option und dem Argument stehen, man kann also auch -o Archive.dfy schreiben.

Insgesamt ist Diffy schwatzhafter geworden und ich habe überall ein bißchen an der Ausgabe gefeilt.

Einige Änderungen bei der Option -x: Beim rekursiven diffen werden jetzt immer Pfade ins Archiv aufgenommen, ansonsten nur dann, wenn die Option -x angegeben ist.

1.23 wie benutze ich diffy

Zuerst einmal: Am bequemsten läßt sich Diffy natürlich von einer Packershell aus benutzen. Wer so eine Shell hat, die Diffy unterstützt oder die für verschiedene Packer konfiguriert werden kann, der sollte die am besten auch benutzen.

Die Amiga-Version ist ein reines CLI-Tool und läßt sich daher von der Workbench aus nur mit dem Execute-Kommando starten. In dem dann erscheinenden Dialog muß dann die Kommandozeile eingegeben werden.
Auf dem Atari gibt es diese Unterscheidung so nicht und Diffy läßt sich auch von einem Desktop aus starten, indem man Verzeichnisse und/oder Dateien auf Diffy zieht. Das einzigste Problem ist, daß man nicht davon ausgehen kann, daß die Dateien in der gewünschten Reihenfolge übergeben werden. Optionen anzugeben ist so auch nicht möglich. Die Standardkonfiguration von Diffy ist aber extra so ausgelegt, daß dies möglichst wenig problematisch wird und Diffy das Dateidatum benutzt, um herauszufinden, welches die alte und welches die neue Datei ist.

Damit man Diffy überhaupt benutzen kann, ist es natürlich absolut notwendig, daß man auch immer eine alte Version aufhebt. Außerdem sollte man immer im Kopf haben, welche Version die Person hat, der

man ein Diffy-Archiv giebt und diese Version als alte Version für die Erstellung eines Archives verwenden. Wenn mehrere Personen verschiedene Versionen haben, dann muß man diese alle aufheben, denn ein Diffy-Archiv ist natürlich absolut abhängig von der jeweils alten Version.

Handelt es sich nur um eine einzige Datei, so reicht es, wenn man diese etwa nach file.old oder so kopiert und diese dann aufhebt, während man an dem Original weiterhin Änderungen vornimmt.
Mit "diffy file.old file" erzeugt man dann einen Patch, der file.old auf den Stand von file bringt.

Handelt es sich aber um mehrere Datei, so sollte man für ein späteres diffen immer den ganzen Verzeichnisbaum kopieren und aufheben. Dann kann man beliebige Änderungen an allen Dateien vornehmen, alte Dateien wegwerfen und neue erzeugen, wie man will, und wenn man dann ein Update herausgeben will, erzeugt man einfach ein Diffy-Archiv zwischen diesen beiden Verzeichnissen.
Wichtig ist aber, das alle Dateien in einem einzigen Verzeichnisbaum liegen und das darin keine Dateien liegen, die nicht dazu gehören sollen.
Mit "diffy -e -n dir.old dir" ereugt man dann ein Archiv, mit dem ein Verzeichnis dir.old auf den Stand von dir gebracht werden kann, inklusive neuer und nicht mehr vorhandener Dateien.
In diesem Fall ist es vielleicht auch sicherer, die Option -c (nicht alt und neu tauschen, falls eine Datei aus dir.old neuer sein sollte) anzugeben. Der Aufruf, der zum Vergleichen zweier Verzeichnisse am besten geeignet ist, lautet also:

```
diffy -c -e -n -oarchiv.dfy dir.old dir
```

1.24 das neue headerformat

In der Version 2.0 von Diffy gibt es ein neues, erweitertes Headerformat in Diffy, das einige neue Features erst ermöglicht und durch weitere Prüfsummen die Archive sicherer macht:

Es lassen sich beliebige weitere Headertypen einführen, die von alten Versionen überlesen werden.

Es enthält eine CRC für die neue Datei, so daß Fehler beim Patchen erkannt werden können.

Im Header wird auch eine Prüfsumme über den Header abgelegt, so daß auch dort Fehler erkannt werden können.

Das Datum der neuen Datei wird darin abgelegt und beim Patchen wiederhergestellt.

Das selbe gilt für die Dateiattribute (lesegeschützt, schreibgeschützt, ...).