tool view

Albert Mietus'

patch

toolview

Elke ingenieur gebruikt tools en iedereen heeft zijn eigen voorkeur; vaak gebaseerd op wat er toevallig als eerste beschikbaar was. Zo ook de ICT-ingenieur. Ook hij gebruikt tools; software-tools wel te verstaan. Daarvan zijn er oneindig veel, waardoor kiezen erg moeilijk is. Veel handige tools worden niet gebruikt, eenvoudigweg omdat niet bekend was dat er een tool is, specifiek voor dat probleem.

Deze rubriek stelt degelijke handige tools voor.
Ken je ook een handig tool, neem dan contact op; dan schrijven we er over!

Mail voor meer info:
ALbert dot Mietus at PTS dot nl

door & more

Patch is bedacht door Larry Wall, die later ook perl uitvond.
Op http://www.gnu.org/software/patch is de GNU/Linux versie van patch te vinden. Er zijn vele andere versies. CVS bijvoorbeeld, heeft een patch ingebouwd.

Om patch met de hand te gebruiken, is het belangrijk om de juiste diff opties te gebruiken. 't Volgende werkt goed:
diff -Naur 1A 1B > patches
Hiermee worden alle veranderingen, recursief in één file opgeslagen.
Vaak is het echter verstandiger om niet een file aan te maken, maar een patchfile per
verandering. Dan kan versie 2A selectief gepatched worden.

Als de patches op bovenstaande manier gemaakt zijn, dan kunnen ze verwerkt worden met:
patch -p0 -d 2A < patches

Tot slot: gebruik, bij meerdere patchfiles, alfabetisch gesorteerde filenamen. De volgorde is namelijk belangrijk! In de praktijk zal iedereen eerst 'patch.aa' gebruiken en 'patch.zz' als laatste.

Een programmeur schrijft geen nieuwe code; tenminste niet vaak. Veel vaker wordt bestaande code veranderd. Die veranderingen zijn geregeld belangrijker dan het resultaat. Er zijn dan ook vele programma's gemaakt om twee versies te kunnen '*diff*'*en*'. Wanneer code onderhouden wordt door derden, zoals bij opensource software, moeten we onze eigen veranderingen steeds opnieuw maken. Voor elke nieuwe versie opnieuw. Dit kan je automatiseren, met *diff* en patch. Met patch kun je uit de ene versie en 't verschil, de tweede versie maken!

'Even de code patchen'

Dit is een van de meest gehoorde uitspraken van software ingenieurs. Dan worden enkele statements of enkele regels code handmatig veranderd en is er weer een bug verholpen. Dit werkt prima als het af en toe gebeuren moet. Maar soms wordt het vervelend. Bijvoorbeeld om herhaaldelijk dezelfde regels te veranderen. Dan zou je willen dat het automatisch kon.

OpenSource bijwerken

Soms, bijvoorbeeld als je broncode van anderen gebruikt, moet je regelmatig dezelfde patches maken. Omdat er regelmatig nieuwe versies van die software uitkomen. Die moet telkens weer aangepast worden aan de locale situatie. De eerste keer is dat leuk werk. De tweede keer is het veel werk. En daarna is het meer van hetzelfde.
Dit gebeurt natuurlijk heel veel bij opensource code. Dat is code die primair onderhouden wordt door de internetgemeenschap en vaak aangepast moet worden om embedded te kunnen gebruiken. Er zijn dan twee bronnen. Als we dat negeren en de code gewoon opslaan, kan het moeizaam worden om over te gaan op een nieuwe versie. Soms wordt er gebruik gemaakt van een versiebeheersysteem dat hiervoor speciale opties heeft (zoals CVS), maar het kan ook met patch!
Patch lijkt op *diff*, maar werkt omgekeerd.

Oud + (Nieuw – Oud) = Nieuw

Wiskundig is het volkomen triviaal. Het verschil tussen twee getallen opgeteld bij één van die getallen is het andere getal. Dat geldt niet alleen voor getallen, maar voor heel veel zaken. Ook voor tekstfiles bijvoorbeeld.

Met *diff* kunnen we het verschil zien tussen twee versies. Als we dat verschil plus de originele versie aan patch geven, kan de andere versie berekend worden. Hiermee kunnen we dus automatisch onze patches aanbrengen.

In de praktijk werkt dit niet alleen op het origineel en onze patches, maar kunnen die specifieke veranderingen ook automatisch verwerkt worden in een nieuwe algemene versie!

Bewaar de verschillen

Om automatisch te kunnen patchen, is een stappenplan nodig. Eerst moet de originele versie ('1A') bewaard worden, waarna we aan de slag gaan om de code te verbeteren. Als dat werkt, hebben we versie '1B'. Met *diff* bepalen we de verschillen tussen 1A en 1B. Die codeflarden slaan we op.
Als er later een nieuwe versie beschikbaar komt ('2A'), kan patch automatisch versie 2B genereren door die bewaarde codeflarden 'op te tellen' bij versie 2A.

Natuurlijk gaat het niet altijd goed. Maar het gaat veel vaker goed dan fout! Als ingenieur kun je je dan richten op de interessante gevallen. Laat de computer de saaie patches maar doen.