EXAMEN: **Scriptingtalen**

1e Bachelor Informatica
prof. dr. Peter Dawyndt
groep 1

donderdag 19-08-2010, 8:30
academiejaar 2009-2010
tweede zittijd

Opgave 1

Gevraagd wordt om een `bash` shell script `pakman` te schrijven. De werking ervan wordt geïllustreerd aan de hand van onderstaande `genix` sessie. Aan het shell script moeten vier argumenten doorgegeven worden (voor deze opgave is het niet nodig om de geldigheid van de argumenten te testen). Het eerste argument is de naam van een Unix commando. De laatste drie argumenten geven respectievelijk een pagina-, regel- en woordnummer aan. Het shell script moet uit de *manual page* van het opgegeven commando het woord ophalen dat op de gegeven positie staat van de gegeven regel op de gegeven pagina. Zo zien we bijvoorbeeld dat `impossible` het 6^e woord is op de 22^e regel van de 35^e pagina van de manual page van het commando `gcc`. Uiteraard is dit afhankelijk van de instelling van de omgevingsvariabele `MANPATH`, waarvan we de waarde voor de voorbeeldsessie ook hebben uitgeschreven.

```
$ pakman bison 4 25 7
this
$ pakman finger 3 9 7
is
$ pakman gcc 75 27 3
impossible
$ echo $MANPATH
/usr/local/man:/usr/man:/opt/csw/man:/opt/sfw/man:/usr/sfw/share/man/
```

Om een gepagineerde versie van manual pages te genereren, moet het shell script `pakman` onderstaande commandolijn gebruiken (hier geïllustreerd voor de manual page van het commando `gcc`). Hierbij zorgt het commando `gsed` voor het verwijderen van controlekarakters uit de manual page, en zorgt het commando `pr` voor de paginering. In de uitvoer kan men het begin van elke pagina herkennen als een regel die eindigt met de tekst `Page` *nnn*, waarbij *nnn* het paginanummer voorstelt. Deze regel vormt meteen ook de eerste regel van de pagina. Woorden worden op een regel van elkaar gescheiden door witruimte (spaties en tabs).

```
man gcc 2> /dev/null | gsed -e 's/.\x08//g' | pr
```

De uitvoer van dit commando moet vervolgens via een pipeline doorgestuurd worden naar het commando `gawk`, dat via een `awk`-script (volledig ingebed binnen het shell script `pakman`) het gevraagde woord uitknipt en uitschrijft naar standaard uitvoer (gevolgd door een *newline*).

Opmerking: Als je het commando `pakman` correct hebt geïmplementeerd, dan krijg je door het meegeleverde commando `geheim` uit te voeren een boodschap te zien die verborgen ligt in de manual pages van `genix`.

Opgave 2

De term *magic number* wordt gebruikt voor een constante bit- of bytesequentie die altijd op dezelfde positie voorkomt binnen bestanden van een bepaald bestandsformaat. Deze magic numbers kunnen daardoor gebruikt worden om het bestandsformaat van een gegeven bestand te achterhalen. Dit is het principe waarop het `file` commando in Unix gebaseerd is.
Gevraagd wordt om een `bash` shell script `findmagic` te schrijven, dat kan gebruikt worden om de magic numbers van een bestandsformaat te achterhalen. Aan dit script moeten minstens twee gewone

bestanden als argument meegegeven worden. Het script moet de lijst van byteposities die identiek zijn voor alle gegeven bestanden naar standaard uitvoer schrijven. Elke regel van deze lijst bestaat uit het volgnummer van een identieke byte (geïndexeerd vanaf 0; offset ten opzichte van het begin van het bestand), gevolgd door een dubbelpunt, een spatie en de waarde van de byte zelf. Afdrukbare karakters moeten als dusdanig uitgeschreven worden, terwijl niet-afdrukbare karakters (ASCII-waarde kleiner dan 32 of gelijk aan 127) in hexadecimale vorm moeten uitgeschreven worden: `\0x`*`HH`* (waarbij *`HH`* twee hexadecimale cijfers zijn). Het maximaal aantal bytes dat door het shell script moet onderzocht worden, kan als argument van de optie `-b` meegegeven worden. Indien de optie `-b` niet gebruikt wordt, dan worden hoogstens de eerste 100 bytes onderzocht. Onderstaande sessie illustreert de werking van het shell script `findmagic`. Hieruit kan afgeleid worden dat men PDF bestanden kan herkennen aan het feit dat de eerste 5 bytes van dergelijke bestanden de tekst `%PDF-` bevatten.

```
$ findmagic *.pdf
0: %
1: P
2: D
3: F
4: -
5: 1
6: .
9: %
$ findmagic -b 5 *.pdf
0: %
1: P
2: D
3: F
4: -
$ findmagic -b 10 *.jpg
0: \0xFF
1: \0xD8
2: \0xFF
3: \0xE0
4: \0x0A
5: \0x10
6: J
7: F
8: I
9: F
$
```

Het shell script `findmagic` moet de geldigheid van de argumenten nagaan, namelijk minstens twee argumenten en uitsluitend bestaande gewone bestanden. Ook de geldigheid van de meegeleverde opties moet nagegaan worden. Het shell script moet een gepaste foutboodschap uitschrijven indien niet aan deze voorwaarden voldaan wordt. Dit wordt geïllustreerd in onderstaande sessie.

```
$ findmagic test1.pdf
findmagic: at least two regular files should be passed as an argument.
Usage: findmagic [-b size] file file ...
$ findmagic test1.pdf test2.pdf test3.pdf /etc test4.pdf
findmagic: /etc is not a regular file.
Usage: findmagic [-b size] file file ...
$ findmagic -x *.pdf
Usage: findmagic [-b size] file file ...
$
```

Tip: Om een byte op een bepaalde positie in een bestand uit te lezen, en om deze byte om te zetten naar zijn decimale of hexadecimale ASCII-waarde, kunnen de volgende commando's gebruikt worden.

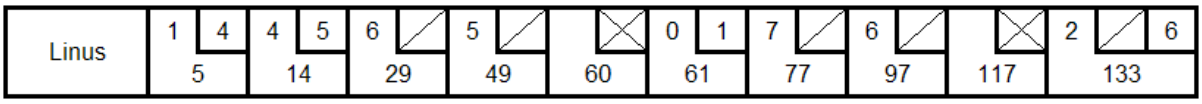
```
# byte op positie p (offset tov begin) in gegeven bestand uitschrijven
dd if=<bestand> bs=1 count=1 skip=<p> 2> /dev/null

# decimale ASCII-waarde van byte "A" uitschrijven
echo "A" | od -td1 | awk 'NR==1{print $2}'

# hexadecimale ASCII-waarde van byte "A" uitschrijven
echo "A" | xxd -u -l1 | awk '{print $2}'
```

Opgave 3

Een spelletje bowling wordt gespeeld over tien frames en heeft een scoreverloop zoals geïllustreerd in onderstaande figuur. Binnen elk frame heeft een speler twee beurten om tien kegels omver te werpen. De score voor een frame is het totaal aantal omvergeworpen kegels, plus een extra bonus voor *strikes* en *spares*. Deze score wordt opgeteld bij de score van het voorgaande frame.



Wanneer een speler alle tien de kegels kan omverwerpen in twee beurten, dan spreken we van een *spare*. De bonus voor dat frame is het aantal kegels dat wordt omvergeworpen bij de eerstvolgende worp. In frame 3 van het voorbeeldspelletje dat hierboven staat weergegeven, bedraagt de score dus 10 (het totaal aantal omvergeworpen kegels) plus een bonus van 5 (het aantal omvergeworpen kegels bij de volgende worp).
Wanneer een speler bij de eerste worp van een frame alle kegels kan omverwerpen, dan spreken we van een *strike*. Bij een strike blijft het aantal worpen in het frame beperkt tot één. De bonus voor dat frame is het totaal aantal kegels dat omvergeworpen wordt bij de volgende twee worpen.
Indien een speler in het tiende frame een spare of strike werpt, dan mag hij extra ballen blijven werpen om het frame af te werken. Er mogen echter nooit meer dan drie ballen geworpen worden in het tiende frame. In het tiende frame worden geen bonuspunten toegekend.
Gegeven is een Excel werkmap `bowling.xlsm`. Daarin bevat het werkblad `bowling` de worpen die een aantal spelers hebben gegooid bij een spelletje bowling. Gevraagd wordt om voor deze spelletjes bowling een grafische voorstelling van het scorebord te genereren op een nieuw werkblad. Hiervoor ga je als volgt te werk:

1. In de module `bowling` werd reeds een VBA subprocedure `toon_frame` geïmplementeerd. Deze procedure schrijft de grafische voorstelling van één enkel frame uit naar een gegeven werkblad. Een frame wordt op twee opeenvolgende rijen uitgeschreven: op de bovenste rij staat het aantal omvergeworpen kegels en op de onderste rij de totaalscore na dit frame. Aan deze subprocedure moeten de volgende argumenten meegegeven worden:

   `ws` een `Worksheet` object dat aangeeft op welk werkblad het frame moet uitgeschreven worden

   `rij` nummer van de rij op het werkblad waarin de worpen moeten geplaatst worden

   `kolom` nummer van de kolom op het werkblad waarin de eerste worp moet geplaatst worden

   `worp` array met het aantal omvergeworpen kegels in alle beurten van het frame; array moet lengte 2 (gewoon frame) of lengte 3 (laatste frame) hebben, en wordt geïndexeerd vanaf 1

   `score` score voor het frame

2. Gebruik de subprocedure `toon_frame` om in VBA een subprocedure `toon_spel` te schrijven, die de grafische voorstelling van een spelletje bowling (voor één enkele speler) uitschrijft naar een gegeven werkblad. Aan deze subprocedure moeten de volgende argumenten meegegeven worden:

   `ws` een `Worksheet` object dat aangeeft op welk werkblad het spel moet uitgeschreven worden

   `rij` nummer van de rij op het werkblad waarin de worpen moeten geplaatst worden

   `kolom` nummer van de kolom op het werkblad waarin de naam van de speler moet geplaatst worden

   `naam` naam van de speler

   `worpen` tekenreeks bestaande uit 21 getallen die van elkaar worden gescheiden door komma's; elk opeenvolgend paar (of drietal in het laatste frame) getallen stelt het aantal omvergeworpen kegels voor in een frame; waarde nul (`0`) geeft aan dat er in deze beurt geen kegels werden

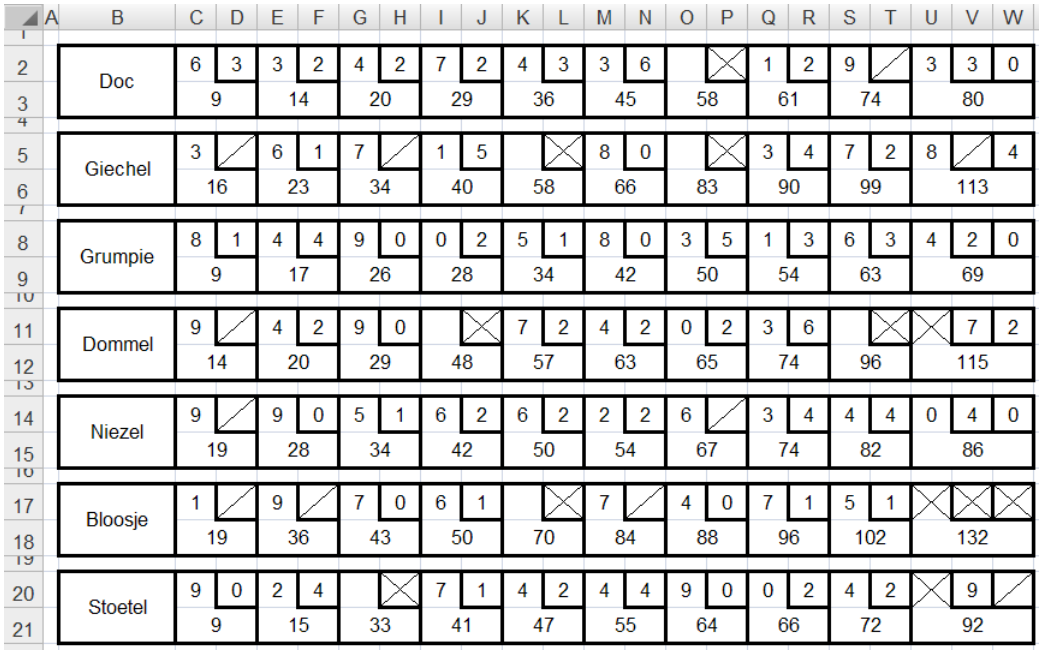
omvergeworpen **of** dat er in deze beurt niet meer moest geworpen worden (bij strike of in laatste frame); onderstaande tekenreeks correspondeert met het voorbeeld dat hierboven werd weergegeven

`1,4,4,5,6,4,5,5,10,0,0,1,7,3,6,4,10,0,2,8,6`

Hierbij geeft de eerste nul aan dat er in het vijfde frame geen tweede beurt was, aangezien alle kegels reeds werden omvergeworpen in de eerste beurt van het frame. De tweede nul geeft aan dat er bij de eerste beurt van het zesde frame geen kegels werden omvergeworpen.

Deze subprocedure moet twee verticaal aangrenzende cellen op het gegeven werkblad samenvoegen, daar een dikke rand rond trekken, en er de naam van de speler in wegschrijven. De breedte van de kolom waarin deze samengevoegde cellen staan moet ingesteld worden op 10. De tien frames moeten uitgeschreven worden rechts van de samengevoegde cel die de naam bevat. Hierbij moet de subprocedure ook het correcte scoreverloop van het spelletje bowling berekenen.

3. Schrijf in VBA een subprocedure `toon_scorebord`, die voor alle spelers op het werkblad `bowling` de grafische voorstelling van het scorebord onder elkaar uitschrijft op een nieuw werkblad (die door de procedure zelf aangemaakt wordt). De eerste naam moet in de cel op de tweede rij en de tweede kolom geplaatst worden, en tussen het scorebord van twee spelletjes moet een lege regel gelaten worden. Zorg ervoor dat de procedure correct blijft werken als er spelers uit de lijst worden verwijderd of eraan worden toegevoegd. Het eindresultaat moet er dan voor de gegeven spelletjes als volgt uitzien:



Opgave 4

Voor Unix hebben extensies in bestandsnamen geen enkele betekenis. Het `file` commando probeert daarentegen het bestandsformaat van bestanden te achterhalen op basis van hun inhoud.

```
$ file logo.png
logo.png:       PNG image data
```

Hiervoor maakt het `file` commando onder andere gebruik van het bestand `/etc/magic`. Dit bestand omschrijft een reeks constante bytepatronen die op een vaste positie binnen een bestandsformaat voorkomen en waaraan de bestanden van dit formaat dus kunnen herkend worden. Vandaar dat men voor

dergelijke patronen de term *magic numbers* gebruikt. Een PNG bestand is bijvoorbeeld te herkennen aan de tekenreeks `\0211PNG` die helemaal aan het begin van het bestand voorkomt. Dit patroon wordt in het bestand `/etc/magic` omschreven door de volgende regel.

```
0       string          \0211PNG        PNG image data
```

Sommige patronen kunnen nog verder gespecifieerd worden, bijvoorbeeld om verschillende versies van een bestandsformaat te onderscheiden. Dit wordt in het bestand `/etc/magic` omschreven door een aantal opeenvolgende regels.

```
0       string          FlAsH-aRcHiVe   Flash Archive
>13     string          -1.0             1.0
>13     string          -2.0             2.0
```

De eerste regel (noem dit het *autonome* patroon) legt vast met welke tekenreeks een `Flash Archive` bestand begint. Bestanden die aan dit patroon voldoen, kunnen verder geanalyseerd worden met twee bijkomende patronen in de daaropvolgende regels (deze starten altijd met een `>` teken): staat 13 karakters voorbij het begin van het bestand de tekenreeks `-1.0` resp. `-2.0`, dan is het bestandsformaat `Flash Archive 1.0` resp. `Flash Archive 2.0`. Is er geen overeenkomst met één van de bijkomende patronen, dan is het bestandsformaat eenvoudigweg `Flash Archive`.
Voor deze opgave is het de bedoeling om het bestand `/etc/magic` om te zetten naar een sed-script `magic.sed`, dat dan op de volgende manier kan toegepast worden.

```
$ gsed -f makemagic.sed /etc/magic > magic.sed
$ gsed -f magic.sed logo.png
PNG image data
```

Om het bestand `magic.sed` automatisch te laten genereren, moet je dus een `sed`-script `makemagic.sed` schrijven dat kan inwerken op het bestand `/etc/magic`. Hieronder wordt stap voor stap beschreven hoe je dit best kunt aanpakken.

1. Voor de eenvoud worden enkel de autonome patronen uit het bestand `/etc/magic` beschouwd die werken met het gegevenstype `string` (2^e kolom). Alle andere patronen uit het bestand `/etc/magic` moeten genegeerd worden. Ook de lege regels en de commentaarregels (die starten met een hekje (`#`)) moeten genegeerd worden.

2. Elk autonoom `string`-patroon moet omgezet worden naar een `sed`-blok dat enkel door `sed` wordt uitgevoerd als de tekst in de *pattern space* overeenkomt met de signatuur van het patroon. In dat `sed`-blok staan twee dingen:

   (a) een groep opdrachten die een vaste tekst wegschrijven naar de *hold buffer*, zonder dat daarbij de inhoud van de *pattern space* gewijzigd wordt

   (b) een herkenbaar stuk commentaar: het ankerpunt

   Deze omzetting wordt verduidelijkt aan de hand van een voorbeeld. Stel dat we het autonome `string`-patroon moeten verwerken dat correspondeert met de volgende regel in `/etc/magic`.

   ```
   17      string          ABRA            MAGISCH BESTAND
   ```

   Dit patroon moet omgezet worden naar het volgende `sed`-blok

   ```
   /<staat tekst "ABRA" op positie 17?>/ {
     <plaats tekst "MAGISCH BESTAND" in hold buffer>        # pattern space behouden !!
   #XXX
   }
   ```

Hierbij hebben we de tekst `#XXX` gebruikt als ankerpunt (het hekje in de eerste kolom van het ankerpunt zorgt er immers voor dat deze regel door `sed` als commentaar aanzien wordt).

Sommige patronen in het bestand `/etc/magic` werken met tekenreeksen die niet-afdrukbare bytes bevatten. Daarom worden ze gecodeerd in decimale of octale vorm. Om met dergelijke tekenreeksen te kunnen werken in `sed`-expressies, moeten de volgende omzettingen gebeuren:

- `\0`*nnn* (octaal; `0` = nul; *nnn* = reeks cijfers) moet worden omgezet naar `\o`*nnn*
- `\`*nnn* (decimaal; *nnn* = reeks cijfers) moet worden omgezet naar `\d`*nnn*

De uitvoer van `gsed -f makemagic.sed /etc/magic > magic.sed` moet nu reeds een geldig `sed`-script zijn. Testen is echter nog niet mogelijk, aangezien de volgende stukken nog ontbreken.

(a) Als eerste opdracht in het `magic.sed` script moet er een controlelus geplaatst worden die het invoerbestand volledig inleest in de *pattern space*. Dit kan eenvoudig met het `N` commando, maar er zijn ook andere mogelijkheden. Maak echter in géén enkel geval gebruik van de *hold buffer*[1].

(b) Na die inleiding worden de geconstrueerde `sed`-blokken losgelaten op de *pattern space*. Patronen die matchen zorgen ervoor dat de uit te schrijven tekst op de *hold buffer* geplaatst wordt. Er moet dus op het einde van het `magic.sed` script een stuk code komen dat ervoor zorgt dat de inhoud van de *hold buffer* wordt uitgeschreven.

Dit levert je een werkende versie op van het `sed` script `makemagic.sed`, die evenwel alleen nog maar rekening houdt met autonome `string`-patronen. Het script kan verder uitgebreid worden om rekening te houden met bijkomende patronen. Dit wordt beschreven in de volgende stap.

3. Elk autonoom `string`-patroon kan een willekeurig aantal bijkomende patronen hebben. Dit zijn regels in het bestand `/etc/magic` die starten met een `>` teken, gevolgd door een reeks cijfers. Deze bijkomende patronen kunnen op de volgende manier verwerkt worden.

   (a) Voeg na het omzetten van de autonome regel, de volgende regel uit het bestand `/etc/magic` toe aan de *pattern space*. Er zijn drie mogelijkheden die verder moeten onderzocht worden.

   (b) De nieuw ingelezen regel is géén bijkomend patroon: zorg ervoor dat het volledige patroon dat zojuist werd omgezet, wordt uitgeschreven naar standaard uitvoer. Het resterend gedeelte van de *pattern space* moet verwerkt worden om te kijken of het een nieuw autonoom `string`-patroon is: spring dus terug naar het begin van het script.

   (c) De nieuw ingelezen regel is een bijkomend patroon, maar werkt niet met het gegevenstype `string`: negeer dit patroon en spring terug naar (a).

   (d) De nieuw ingelezen regel is een bijkomend `string`-patroon: zet dit bijkomend patroon om naar een `sed`-blok dat een vaste tekst toevoegt aan de *hold buffer*. Opnieuw mag hierbij de bestaande inhoud van de *pattern space* niet gewijzigd worden! Plaats dit blok in het autonome blok net voor het ankerpunt. Let er echter wel op dat het ankerpunt behouden blijft. Spring na deze stap terug naar (a) om de andere bijkomende patronen te verwerken, indien er nog zijn.

Volgens bovenstaande procedure, wordt de volgende groep van patronen

```
17      string          ABRA            MAGISCH BESTAND
>21     string          CADA            : behold
>25     string          BRA              the magic
```

[1] Sommige `sed`-implementaties *interpreteren* controlekarakters in de *hold buffer*, wat problemen kan opleveren met binaire bestanden. Hou deze binaire bestanden dus weg uit de *hold buffer*, tenzij om deze op zijn geheel om te wisselen met de inhoud van de pattern space.

omgezet naar het volgende `sed`-blok.

```
/<staat tekst "ABRA" op positie 17?>/ {
  <plaats tekst "MAGISCH BESTAND" in hold buffer>        # pattern space behouden !!
  /<staat tekst "CADA" op positie 21?>/ {
    <voeg tekst ": behold" toe aan hold buffer>          # pattern space behouden !!
  }
  /<staat tekst "BRA" op positie 25?>/ {
    <voeg tekst " the magic" toe aan hold buffer>        # pattern space behouden !!
  }
#XXX
}
```

Tips:

- met `man magic` kan je de omschrijving van het bestandsformaat van `/etc/magic` opvragen
- substituties in `sed`: gebruik bijvoorbeeld `s|a|b|` of `s@a@b@` om minder te moeten escapen
- reguliere expressies: denk aan accolades om een vast aantal herhalingen aan te geven
- op `genix`: gebruik de GNU-versie `gsed` van `sed` in `/opt/csw/bin/`