Objectives

- Practice writing shell scripts for real tasks.
- Practice processing collections of files with shell scripts.

Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples.

Getting Started

Create a new directory for this lab called `lab04` by typing:

```
$ mkdir lab04
```

Change to this directory by typing:

```
$ cd lab04
```

Exercise: Converting Images from JPG to PNG

Write a shell script `jpg2png.sh` which converts all images in JPEG format in the current directory to PNG format.
You can assume that JPEG files and only JPEG files have the suffix `jpg`.

If the conversion is succesful the JPEG file should be removed.

Your script should stop with the error message shown below and exit status 1 if the PNG file already exists.

```
$ wget https://cgi.cse.unsw.edu.au/~cs2041/18s2//activities/jpg2png/images.zip
$ unzip images.zip
Archive:  images.zip
  inflating: Johannes Vermeer - The Girl With The Pearl Earring.jpg
  inflating: nautilus.jpg
  inflating: panic.jpg
  inflating: penguins.jpg
  inflating: shell.jpg
  inflating: stingray.jpg
  inflating: treefrog.jpg
$ ./jpg2png.sh
$ ls
'Johannes Vermeer - The Girl With The Pearl Earring.png'   jpg2png.sh
     panic.png  shell.png      treefrog.png
 images.zip                        nautilus.png   penguins.png   stin
gray.png
$ wget https://cgi.cse.unsw.edu.au/~cs2041/18s2//activities/jpg2png//penguins.jpg
$ ls
'Johannes Vermeer - The Girl With The Pearl Earring.png'   jpg2png.sh
     panic.png  penguins.png   stingray.png
 images.zip                        nautilus.png   penguins.jpg   shel
l.png      treefrog.png
$ ./jpg2png.sh
penguins.png already exists
```

Hints

You may find sed and back quotes useful.

The tool `convert` will convert between many image formats, for example:

```
$ convert penguins.jpg penguins.png
```

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest jpg2png
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab04_jpg2png jpg2png.sh
```

Sample solution for `jpg2png.sh`

```
#!/bin/sh

for jpg_file in *.jpg
do
    png_file=`echo "$jpg_file"|sed 's/jpg$/png/'`
    if test -e "$png_file"
    then
        echo "$png_file" already exists
        exit 1
    fi
    convert "$jpg_file" "$png_file" && rm "$jpg_file"
done
```

Exercise: Email that Image?

Write a shell script `email_image.sh` which given a list of image files as arguments displays them one-by-one. After the user has viewed each image the script should prompt the user for an e-mail address. If the user does enter an email address, the script should prompt the user for a message to accompany the image and then send the image to the specified e-mail address.

```
$ ./email_image.sh penguins.png treefrog.png
Address to e-mail this image to? andrewt@cse.unsw.edu.au
Message to accompany image? Penguins are cool.
penguins.png sent to andrewt@cse.unsw.edu.au
Address to e-mail this image to? andrewt@cse.unsw.edu.au
Message to accompany image? This is a White-lipped Tree Frog
treefrog.png sent to andrewt@cse.unsw.edu.au
```

Hints

The program `display` can be used to view image files

The program `mutt` can be used to send mail from the command line including attachments, for example:

```
$ echo 'Penguins are cool.'|mutt -s 'penguins!' -e 'set copy=no' -a penguins.png -- nobody@nowhere.com
```

For comparison to the Shell a Python solution

```
#!/usr/bin/python3

import smtplib, subprocess, sys
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

from_address = "andrewt@unsw.edu.au"
for png_file in sys.argv[1:]:
    subprocess.check_output(['display', png_file])
    sys.stdout.write("Address to e-mail this image to? ")
    sys.stdout.flush()
    to_address = sys.stdin.readline().strip()
    if to_address:
        sys.stdout.write("Message to accompany image? ")
        sys.stdout.flush()
        message = sys.stdin.readline().strip()
        msg = MIMEMultipart(message)
        msg['Subject'] = png_file
        msg['From'] = from_address
        msg['To'] = to_address
        with open(png_file) as f:
            attachment = MIMEText(f.read())
            attachment.add_header('Content-Disposition', 'attachment', filename=png_file)
            msg.attach(attachment)
        s = smtplib.SMTP('smtp.cse.unsw.edu.au')
        s.sendmail(from_address, [to_address], msg.as_string())
        s.quit()
    else:
        print("No email sent")
```

There is no autotest and no automarking of this question.
When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab04_email_image email_image.sh
```

Sample solution for `email_image.sh`

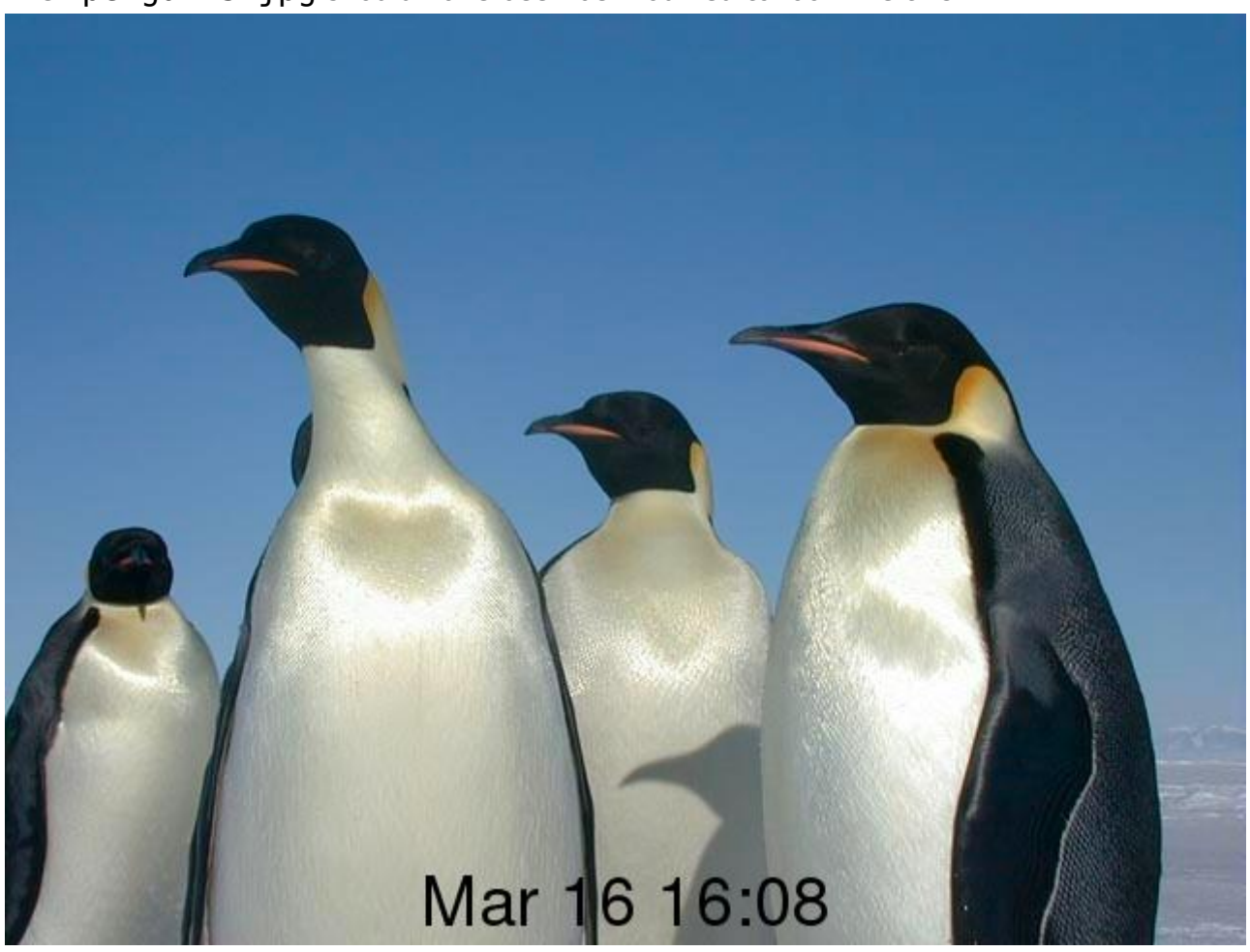
```
#!/bin/sh

for png_file in "$@"
do
    display "$png_file"
    echo -n "Address to e-mail this image to? "
    read address
    if test -n "$address"
    then
        echo -n "Message to accompany image? "
        read message
        echo "$message"| mutt -s 'image' -a "$png_file" -e 'set copy=no' -- "$address"
        echo "$png_file sent to $address"
    else
        echo "No email sent"
    fi
done
```

Exercise: Date A Penguin?

Write a shell script `date_image.sh` which, given a list of image files as arguments, changes each file so it has a label added to the image indicating the time it was taken. You can assume the last-modification time of the image file is the time it was taken. So for example if we run these commands:

```
$ cp -p /web/cs2041/18s2/activities/date_image/penguins.jpg .
$ ls -l penguins.jpg
-rw-r--r-- 1 andrewt andrewt 58092 Mar 16 16:08 penguins.jpg
$ ./date_image.sh penguins.jpg
$ display  penguins.jpg
```

Then `penguins.jpg` should have been be modified to look like this:



Hints

The program `convert` can be used to label an image like this:

```
$  convert -gravity south -pointsize 36 -draw "text 0,10 'Andrew rocks'" penguins.jpg temporary_file.jpg
```

Hint: `sed` and/or `cut` may be useful to extract the date & time from ls's output.

Hint: `convert` produce confusing messages if you don't get its option syntax exactly right There is no autotest and no automarking of this question.

When you have completed demonstrate your work to another student in your lab and ask them to enter a peer assessment here

It is prefered you do this during your lab, but if this is not possible you may demonstrate your work to any other COMP[29]041 student before Sunday midnight. Note, you must also submit the work with give.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab04_date_image date_image.sh
```

Sample solution for `date_image.sh`

```
#!/bin/sh

for image_file in "$@"
do
    last_modify_time=`ls -l "$image_file"|cut -d\  -f5-|sed 's/^ *//'|cut -d\  -f2-4`
    temporary_file="$image_file.tmp.$$"
    if test -e "$temporary_file"
    then
        echo "$temporary_file" already exists
        exit 1
    fi
    convert -gravity south -pointsize 36 -draw "text 0,10 '$last_modify_time'" "$image_file" "$tempor
    touch -r "$image_file" "$temporary_file" &&  # preserve modification time (challenge question)
    mv "$temporary_file" "$image_file"
done
```

Exercise: Tagging a Collection of Music

Andrew regularly spends time far from the internet and streaming music services such as Spotify, so he has a large collection of MP3 files containing music.
Andrew has a problem - the ID3 tags in the MP3 files in his music collection are incorrect. Unfortunately Andrew's favourite player software organizes music using the information from these ID3 tags. Your task it to fix Andrew's problem by set the ID3 tags to the correct values. Fortunately the correct value for the tags can be retrieved from the file names and the names of the directories the files are in.

Your task is to write a shell script `tag_music.sh` which sets the ID3 tags of MP3 files using the information from file names and directory names.

You'll first need to make a copy of Andrew's music collection.

Download music.zip here, or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/18s2/activities/tag_music/music.zip .
```

You assume the names of files and directories follow a standard format. You can determine this format by look at ethe files in Andrew's music collection.

```
$ unzip music.zip
Archive:  music.zip
   creating: music/
   creating: music/Triple J Hottest 100, 2007/
  inflating: music/Triple J Hottest 100, 2007/2 - Straight Lines - Si
lverchair.mp3
  inflating: music/Triple J Hottest 100, 2007/10 - Don't Fight It - T
he Panics.mp3
 ...
```

The command `id3` can be used to list the value of ID3 tags in an MP3 file, for example:

```
$ id3 -l 'music/Triple J Hottest 100, 2013/1 - Riptide - Vance Joy.mp3'
music/Triple J Hottest 100, 2013/1 - Riptide - Vance Joy.mp3:
Title  : Andrew Rocks                    Artist: Andrew
Album  : Best of Andrew                  Year: 2038, Genre: Unknown
 (255)
Comment:                                 Track: 42
```

As you can see the ID3 tags of this music file have been accidentally over-written. The ID3 tags should be:

```
$ id3 -l 'music/Triple J Hottest 100, 2013/1 - Riptide - Vance Joy.mp3'
music/Triple J Hottest 100, 2013/1 - Riptide - Vance Joy.mp3:
Title  : Riptide                          Artist: Vance Joy
Album  : Triple J Hottest 100, 2013      Year: 2013, Genre: Unknown
 (255)
Comment:                                 Track: 1
```

Fortunately all the information needed to fix the ID3 tags is available in the name of the file and the name of the directory it is in. You will write a shell script `tag_music.sh` which takes the name of 1 or more directories as arguments and fixes the ID3 tags of the all MP3 files in that directory. For example:

```
$ ./tag_music.sh 'music/Triple J Hottest 100, 2015'
$ id3 -l 'music/Triple J Hottest 100, 2015/4 - The Less I Know the Better - Tame Impala.mp3'
music/Triple J Hottest 100, 2015/4 - The Less I Know the Better - Tam
e Impala.mp3:
Title  : The Less I Know the Better      Artist: Tame Impala
Album  : Triple J Hottest 100, 2015      Year: 2015, Genre: Unknown
 (255)
Comment:                                 Track: 4
$ ./tag_music.sh music/*
$ id3 -l 'music/Triple J Hottest 100, 1995/10 - Greg! The Stop Sign!! - TISM.mp3'
music/Triple J Hottest 100, 1995/10 - Greg! The Stop Sign!! - TISM.mp
3:
Title  : Greg! The Stop Sign!!           Artist: TISM
Album  : Triple J Hottest 100, 1995      Year: 1995, Genre: Unknown
 (255)
Comment:                                 Track: 10
$ id3 -l 'music/Triple J Hottest 100, 1999/1 - These Days - Powderfinger.mp3'
music/Triple J Hottest 100, 1999/1 - These Days - Powderfinger.mp3:
Title  : These Days                      Artist: Powderfinger
Album  : Triple J Hottest 100, 1999      Year: 1999, Genre: Unknown
 (255)
Comment:                                 Track: 1
$ id3 -l 'music/Triple J Hottest 100, 2012/2 - Little Talks - Of Monsters and Men.mp3'
music/Triple J Hottest 100, 2012/2 - Little Talks - Of Monsters and M
en.mp3:
Title  : Little Talks                    Artist: Of Monsters and Men
Album  : Triple J Hottest 100, 2012      Year: 2012, Genre: Unknown
 (255)
Comment:                                 Track: 2
```

Your script should not change the *Genre* or *Comment* fields.
Your script should determine *Title*, *Artist*, *Track*, *Album* & *Year* from the directory & filename.

Hints

```
$ man id3
...
```

`cut` almost works for extracting *Title* and *Album* from the filename.
Handling the few MP3 files correctly where using `cut` doesn't work will be considered a **challenge exercise**.

It can be difficult debugging your script on Andrew's music collection. In cases like these it usually worth creating a smaller data set for initial debugging. Such a tiny data set is available in tiny_music.zip if you want to use it for debugging. This dataset is used in the first autotests. Download tiny_music.zip here, or copy it to your CSE account using the following command:

```
$ cp -n /web/cs2041/18s2/activities/tag_music/tiny_music.zip .
```

```
$ unzip tiny_music.zip
Archive:  tiny_music.zip
   creating: tiny_music/
   creating: tiny_music/Album1, 2015/
  inflating: tiny_music/Album1, 2015/2 - Little Talks - Of Monsters a
nd Men.mp3
  inflating: tiny_music/Album1, 2015/1 - Riptide - Vance Joy.mp3
   creating: tiny_music/Album2, 2016/
  inflating: tiny_music/Album2, 2016/2 - Royals - Lorde.mp3
  inflating: tiny_music/Album2, 2016/1 - Hoops - The Rubens.mp3
$ id3 -l tiny_music/*/*.mp3
tiny_music/Album1, 2015/1 - Riptide - Vance Joy.mp3:
Title  : Andrew Rocks                    Artist: Andrew
Album  : Best of Andrew                  Year: 2038, Genre: Unknown
 (255)
Comment:                                 Track: 42
tiny_music/Album1, 2015/2 - Little Talks - Of Monsters and Men.mp3:
Title  : Andrew Rocks                    Artist: Andrew
Album  : Best of Andrew                  Year: 2038, Genre: Unknown
 (255)
Comment:                                 Track: 42
tiny_music/Album2, 2016/1 - Hoops - The Rubens.mp3:
Title  : Andrew Rocks                    Artist: Andrew
Album  : Best of Andrew                  Year: 2038, Genre: Unknown
 (255)
Comment:                                 Track: 42
tiny_music/Album2, 2016/2 - Royals - Lorde.mp3:
Title  : Andrew Rocks                    Artist: Andrew
Album  : Best of Andrew                  Year: 2038, Genre: Unknown
 (255)
Comment:                                 Track: 42
$ ./tag_music.sh tiny_music/*
$ id3 -l tiny_music/*/*.mp3
tiny_music/Album1, 2015/1 - Riptide - Vance Joy.mp3:
Title  : Riptide                         Artist: Vance Joy
Album  : Album1, 2015                    Year: 2015, Genre: Unknown
 (255)
Comment:                                 Track: 1
tiny_music/Album1, 2015/2 - Little Talks - Of Monsters and Men.mp3:
Title  : Little Talks                    Artist: Of Monsters and Men
Album  : Album1, 2015                    Year: 2015, Genre: Unknown
 (255)
Comment:                                 Track: 2
tiny_music/Album2, 2016/1 - Hoops - The Rubens.mp3:
Title  : Hoops                           Artist: The Rubens
Album  : Album2, 2016                    Year: 2016, Genre: Unknown
 (255)
Comment:                                 Track: 1
tiny_music/Album2, 2016/2 - Royals - Lorde.mp3:
Title  : Royals                          Artist: Lorde
Album  : Album2, 2016                    Year: 2016, Genre: Unknown
 (255)
Comment:                                 Track: 2
```

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest tag_music
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab04_tag_music tag_music.sh
```

Sample solution for `tag_music.sh`

```
#!/bin/sh

for album_pathname in "$@"
do
    album=`basename "$album_pathname"`
    year=`echo "$album"|sed 's/.* //'`

    for mp3_pathname in "$album_pathname"/*.mp3
    do
        mp3_filename=`basename "$mp3_pathname" .mp3`
        # assume ' - ' doesn't occur in artist or album
        track=`echo "$mp3_filename"|sed 's/ - .*//'`
        title=`echo "$mp3_filename"|sed 's/^[0-9]* - //;s/ - .*//'`
        artist=`echo "$mp3_filename"|sed 's/.* - //'`
        id3 -t "$title" -T "$track" -a "$artist" -A "$album" -y "$year" "$mp3_pathname" >/dev/null
    done
done
```

Challenge Exercise: Creating A Fake Music Collection

The test data for the previous question is not really Andrew's music collection. All the mp3 files contain identical contents. The directories and filenames were created from the source of this web page.

Write a shell script `create_music.sh` which uses the above webpage to create exactly the same directories and files as in the test data set supplied above.

Your script should take 2 arguments: the name of an MP3 file to use as the contents of the MP3 files you create and the directory in which to create the test data. For example:

```
$ wget https://cgi.cse.unsw.edu.au/~cs2041/18s2//activities/create_music/sample.mp3
$ mkdir my_fake_music
$ ls my_fake_music
$ ./create_music.sh sample.mp3 my_fake_music
$ ls my_fake_music
'Triple J Hottest 100, 1993'  'Triple J Hottest 100, 1998'  'Triple J
 Hottest 100, 2003'  'Triple J Hottest 100, 2008'  'Triple J Hottest
 100, 2013'
'Triple J Hottest 100, 1994'  'Triple J Hottest 100, 1999'  'Triple J
 Hottest 100, 2004'  'Triple J Hottest 100, 2009'  'Triple J Hottest
 100, 2014'
'Triple J Hottest 100, 1995'  'Triple J Hottest 100, 2000'  'Triple J
 Hottest 100, 2005'  'Triple J Hottest 100, 2010'  'Triple J Hottest
 100, 2015'
'Triple J Hottest 100, 1996'  'Triple J Hottest 100, 2001'  'Triple J
 Hottest 100, 2006'  'Triple J Hottest 100, 2011'  'Triple J Hottest
 100, 2016'
'Triple J Hottest 100, 1997'  'Triple J Hottest 100, 2002'  'Triple J
 Hottest 100, 2007'  'Triple J Hottest 100, 2012'  'Triple J Hottest
 100, 2017'
$ ls 'my_fake_music/Triple J Hottest 100, 2017'
'1 - Humble - Kendrick Lamar.mp3'                  '5 - The Deepest Sig
hs, the Frankest Shadows - Gang of Youths.mp3'
'10 - What Can I Do If the Fire Goes Out? - Gang of Youths.mp3'  '6 -
 Green Light - Lorde.mp3'
'2 - Let Me Down Easy - Gang of Youths.mp3'          '7 - Go Bang - P
nau.mp3'
'3 - Chateau - Angus & Julia Stone.mp3'              '8 - Sally - Thu
ndamentals featuring Mataya.mp3'
'4 - Ubu - Methyl Ethel.mp3'                       '9 - Lay It on Me -
 Vance Joy.mp3'
$ wget https://cgi.cse.unsw.edu.au/~cs2041/18s2//activities/create_music/music.zip
$ unzip music.zip
...
$ diff -r music my_fake_music
$
```

Hints

```
$ wget -q -O- 'https://en.wikipedia.org/wiki/Triple_J_Hottest_100?action=raw'
...
```

You may find this web page useful for dealing with unicode characters such as *en dash*.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 2041 autotest create_music
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 lab04_create_music create_music.sh
```

Sample solution for `create_music.sh`

```
#!/bin/sh

mp3_file="$1"
base_dir="$2"

wget -q -O- 'https://en.wikipedia.org/wiki/Triple_J_Hottest_100?action=raw'|
while read line
do
    # look for line which is start of Hottest 100 list for a year

    case "$line" in
    *'[[Triple J'*'|'[0-9][0-9][0-9][0-9]']]'*) ;;
    *) continue;;
    esac

    # create a directory for a Hottest 100 year
    album=`echo "$line"|sed 's/.*\[\[//;s/|.*//'`
    year=`echo "$album"|sed 's/.*\ //'`
    dir="$base_dir/Triple J Hottest 100, $year"
    mkdir -p -m 755 "$dir"

    # read top 10 songs for year
    track=1
    while read line && test $track -le 10
    do
        case "$line" in
        '#'*) ;;
        *) continue;;
        esac

        # remove links to wikipedia pages
        line=`echo "$line"|sed 's/[^[]*|//g'`

        # change slashes to hyphens - because can't have / in a filename
        line=`echo "$line"|sed 's/\//-/g'`

        # remove some formating characters
        line=`echo "$line"|tr -d '[]"#'`

        #break line in two at en dash byte codes
        artist=`echo "$line"|sed 's/\xe2\x80\x93.*//'`
        title=`echo "$line"|sed 's/.*\xe2\x80\x93//'`

        #trim leading spaces
        artist=`echo "$artist"|sed 's/^ *//'`
        title=`echo "$title"|sed 's/^ *//'`

        #trim trailing spaces
        artist=`echo "$artist"|sed 's/ *$//'`
        title=`echo "$title"|sed 's/ *$//'`

        file="$dir/$track - $title - $artist.mp3"
        cp -p "$mp3_file" "$file"
        track=$((track + 1))
    done
done
```

Submission

When you are finished each exercises make sure you submit your work by running **give**.
You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Monday 20 August 23:59:59** to submit your work.

You cannot obtain marks by e-mailing lab work to tutors or lecturers.

You check the files you have submitted here

Automarking will be run by the lecturer several days after the submission deadline for the test, using test cases that you haven't seen: different to the test cases `autotest` runs for you.

(Hint: do your own testing as well as running `autotest`)

After automarking is run by the lecturer you can view it here the resulting mark will also be available via via give's web interface

Lab Marks

When all components of a lab are automarked you should be able to view the the marks via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

The lab exercises for each week are worth in total 1 mark.
The best 10 of your 11 lab marks will be summed to give you a mark out of 9. If their sum exceeds 9 - your mark will be capped at 9.

- You can obtain full marks for the labs without completing challenge exercises
- You can miss 1 lab without affecting your mark.