

Files

- archiving files with tar
- concatenation, redirection and pipes

Python

- standard input and output
- the os module
- directory methods

Summary + Assignments

Files

- archiving files with tar
- concatenation, redirection and pipes

Python

- standard input and output
- the os module
- directory methods

Summary + Assignments

MCS 260 Lecture 23
Introduction to Computer Science
Jan Vershelde, 19 October 2007

Files

archiving files with tar

concatenation, redirection and pipes

Files

archiving files with tar

concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Python

standard input and output

the os module

directory methods

Summary + Assignments

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Archiving Files

with the utility `tar`

To create an archive of several files, we use `tar`.

`tar` is an archiving program to store and extract files from an archive, called a *tarfile*. The `t` in `tar` stands for “tape” although we mostly write to a file.

Suppose we want to archive all our Python programs:

1. create a separate directory `pyprogs`;
2. copy all files with extension `.py` to `pyprogs`;
3. apply `tar` (eventually followed by `gzip`).

A session at the Linux command prompt `$`:

```
$ mkdir pyprogs
$ cp *.py pyprogs
$ tar -cf pyprogs.tar pyprogs
```

The `c` and `f` of `tar` are for create and file.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Using File Archives

displaying and extracting files

To display the files in the archive file `pyprogs.tar`:

```
$ tar -tf pyprogs.tar
```

Again, the `f` is for `file`, while `t` is to display.

To extract the files for the archive file `pyprogs.tar`:

```
$ tar -xf pyprogs.tar
```

The `x` and `f` of `tar` are for `extract` and `file`.

If the directory `pyprogs` already exists, then `tar` will update the files with newer versions, otherwise the directory `pyprogs` will be created.

Files

archiving files with tar

concatenation, redirection and pipes

Python

standard input and output

the `os` module

directory methods

Summary + Assignments

Using File Archives

displaying and extracting files

To display the files in the archive file `pyprogs.tar`:

```
$ tar -tf pyprogs.tar
```

Again, the `f` is for `file`, while `t` is to display.

To extract the files for the archive file `pyprogs.tar`:

```
$ tar -xf pyprogs.tar
```

The `x` and `f` of `tar` are for `extract` and `file`.

If the directory `pyprogs` already exists, then `tar` will update the files with newer versions, otherwise the directory `pyprogs` will be created.

Files

archiving files with tar

concatenation, redirection and pipes

Python

standard input and output

the `os` module

directory methods

Summary + Assignments

Using File Archives

displaying and extracting files

To display the files in the archive file `pyprogs.tar`:

```
$ tar -tf pyprogs.tar
```

Again, the `f` is for `file`, while `t` is to display.

To extract the files for the archive file `pyprogs.tar`:

```
$ tar -xf pyprogs.tar
```

The `x` and `f` of `tar` are for `extract` and `file`.

If the directory `pyprogs` already exists, then `tar` will update the files with newer versions, otherwise the directory `pyprogs` will be created.

Files

archiving files with tar

concatenation, redirection and pipes

Python

standard input and output

the `os` module

directory methods

Summary + Assignments

Files

archiving files with tar
concatenation, redirection and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Concatenation of Files

with the utility `cat`

Suppose we want to send small files to the printer.

As all files fit on one page, we better make one new file that contains everything.

At the command prompt `$`

```
$ cat g*.py > toprint
```

This command combines three actions:

1. `cat` concatenates files, given as arguments;
2. `g*.py`: all files starting with `g` and ending with `.py`;
3. instead of writing to screen, with `>` all output goes to the file `toprint`.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Concatenation of Files

with the utility `cat`

Suppose we want to send small files to the printer.

As all files fit on one page, we better make one new file that contains everything.

At the command prompt `$`

```
$ cat g*.py > toprint
```

This command combines three actions:

1. `cat` concatenates files, given as arguments;
2. `g*.py`: all files starting with `g` and ending with `.py`;
3. instead of writing to screen, with `>` all output goes to the file `toprint`.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Concatenation of Files

with the utility `cat`

Suppose we want to send small files to the printer.

As all files fit on one page, we better make one new file that contains everything.

At the command prompt `$`

```
$ cat g*.py > toprint
```

This command combines three actions:

1. `cat` concatenates files, given as arguments;
2. `g*.py`: all files starting with `g` and ending with `.py`;
3. instead of writing to screen, with `>` all output goes to the file `toprint`.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Concatenation of Files

with the utility `cat`

Suppose we want to send small files to the printer.

As all files fit on one page, we better make one new file that contains everything.

At the command prompt `$`

```
$ cat g*.py > toprint
```

This command combines three actions:

1. `cat` concatenates files, given as arguments;
2. `g*.py`: all files starting with `g` and ending with `.py`;
3. instead of writing to screen, with `>` all output goes to the file `toprint`.

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Concatenation of Files

with the utility `cat`

Files

archiving files with `tar`
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Suppose we want to send small files to the printer.

As all files fit on one page, we better make one new file that contains everything.

At the command prompt `$`

```
$ cat g*.py > toprint
```

This command combines three actions:

1. `cat` concatenates files, given as arguments;
2. `g*.py`: all files starting with `g` and ending with `.py`;
3. instead of writing to screen, with `>` all output goes to the file `toprint`.

Redirection and Pipes

sorting misspelled words

`spell` reads text and writes misspelled words.

```
$ spell  
hello  
helo  
helo
```

The first two lines were input, the last line was output.

To print all misspelled words in a the file `sometext`:

```
$ spell < sometext
```

To see a sorted list of misspelled words:

```
$ spell sometext | sort
```

The `|` indicates a **pipe**, setting up a pipeline redirecting the output of `spell` to the input of `sort`, without creating an intermediate file, gluing two different programs.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Redirection and Pipes

sorting misspelled words

`spell` reads text and writes misspelled words.

```
$ spell  
hello  
helo  
helo
```

The first two lines were input, the last line was output.

To print all misspelled words in a the file `sometext`:

```
$ spell < sometext
```

To see a sorted list of misspelled words:

```
$ spell sometext | sort
```

The `|` indicates a **pipe**, setting up a pipeline redirecting the output of `spell` to the input of `sort`, without creating an intermediate file, gluing two different programs.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Redirection and Pipes

sorting misspelled words

`spell` reads text and writes misspelled words.

```
$ spell  
hello  
helo  
helo
```

The first two lines were input, the last line was output.

To print all misspelled words in a the file `sometext`:

```
$ spell < sometext
```

To see a sorted list of misspelled words:

```
$ spell sometext | sort
```

The `|` indicates a **pipe**, setting up a pipeline redirecting the output of `spell` to the input of `sort`, without creating an intermediate file, gluing two different programs.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Pattern Matching

another example of a pipeline

The pipe functionality summarizes the power of Unix.

Another illustration:

```
$ cat *.py | grep "MCS 260" | more
```

is a pipe of three programs:

1. `cat *.py` shows the content of all `.py` files
2. `grep "MCS 260"` prints all lines in the files that match the string "MCS 260"
3. `more` pauses printing after first screen is full.

With Python we can glue several applications *independent* of the operating system.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Pattern Matching

another example of a pipeline

The pipe functionality summarizes the power of Unix.

Another illustration:

```
$ cat *.py | grep "MCS 260" | more
```

is a pipe of three programs:

1. `cat *.py` shows the content of all `.py` files
2. `grep "MCS 260"` prints all lines in the files that match the string "MCS 260"
3. `more` pauses printing after first screen is full.

With Python we can glue several applications *independent* of the operating system.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Pattern Matching

another example of a pipeline

The pipe functionality summarizes the power of Unix.

Another illustration:

```
$ cat *.py | grep "MCS 260" | more
```

is a pipe of three programs:

1. `cat *.py` shows the content of all `.py` files
2. `grep "MCS 260"` prints all lines in the files that match the string `"MCS 260"`
3. `more` pauses printing after first screen is full.

With Python we can glue several applications
independent of the operating system.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Pattern Matching

another example of a pipeline

The pipe functionality summarizes the power of Unix.

Another illustration:

```
$ cat *.py | grep "MCS 260" | more
```

is a pipe of three programs:

1. `cat *.py` shows the content of all `.py` files
2. `grep "MCS 260"` prints all lines in the files that match the string "MCS 260"
3. `more` pauses printing after first screen is full.

With Python we can glue several applications *independent* of the operating system.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Files

archiving files with tar
concatenation, redirection and pipes

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Python

standard input and output
the os module
directory methods

Summary + Assignments

Summary + Assignments

Standard Input and Output

The screen is also considered as a file:

```
>>> import sys
>>> sys.stdout.write('hello there')
hello there>>>
```

Behaves (almost) as `print`.

The input is the standard file `sys.stdin`:

```
>>> name = sys.stdin.readline()
a line given on input
>>> name
'a line given on input\n'
```

Is similar to `raw_input()`.

Error messages can be written to `sys.stderr`.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Standard Input and Output

The screen is also considered as a file:

```
>>> import sys
>>> sys.stdout.write('hello there')
hello there>>>
```

Behaves (almost) as `print`.

The input is the standard file `sys.stdin`:

```
>>> name = sys.stdin.readline()
a line given on input
>>> name
'a line given on input\n'
```

Is similar to `raw_input()`.

Error messages can be written to `sys.stderr`.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Standard Input and Output

The screen is also considered as a file:

```
>>> import sys
>>> sys.stdout.write('hello there')
hello there>>>
```

Behaves (almost) as `print`.

The input is the standard file `sys.stdin`:

```
>>> name = sys.stdin.readline()
a line given on input
>>> name
'a line given on input\n'
```

Is similar to `raw_input()`.

Error messages can be written to `sys.stderr`.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Files

archiving files with tar
concatenation, redirection and pipes

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Python

standard input and output
the os module
directory methods

Summary + Assignments

Summary + Assignments

The os Module

operating system independence

The module `os` is an interface to the operating system.
To execute any command, use `os.system`:

```
>>> import os
>>> os.system('ls')
```

The module `os` exports functions on files and directories,
independent of specific operating system commands.

To list the content of a directory:

```
os.listdir( < directory name > )
```

on return is a list of file names (as strings).

For example, for the current directory:

```
>>> os.listdir('.')
```

Runs on Unix, MacOS X, and Windows: **portable**.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

The os Module

operating system independence

The module `os` is an interface to the operating system.

To execute any command, use `os.system`:

```
>>> import os
>>> os.system('ls')
```

The module `os` exports functions on files and directories, independent of specific operating system commands.

To list the content of a directory:

```
os.listdir( < directory name > )
```

on return is a list of file names (as strings).

For example, for the current directory:

```
>>> os.listdir('.')
```

Runs on Unix, MacOS X, and Windows: **portable**.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

The os Module

operating system independence

The module `os` is an interface to the operating system.

To execute any command, use `os.system`:

```
>>> import os
>>> os.system('ls')
```

The module `os` exports functions on files and directories, independent of specific operating system commands.

To list the content of a directory:

```
os.listdir( < directory name > )
```

on return is a list of file names (as strings).

For example, for the current directory:

```
>>> os.listdir('.')
```

Runs on Unix, MacOS X, and Windows: **portable**.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

The os Module

operating system independence

The module `os` is an interface to the operating system.

To execute any command, use `os.system`:

```
>>> import os
>>> os.system('ls')
```

The module `os` exports functions on files and directories, independent of specific operating system commands.

To list the content of a directory:

```
os.listdir( < directory name > )
```

on return is a list of file names (as strings).

For example, for the current directory:

```
>>> os.listdir('.')
```

Runs on Unix, MacOS X, and Windows: **portable**.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Files

archiving files with tar
concatenation, redirection and pipes

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Python

standard input and output
the os module
directory methods

Summary + Assignments

Summary + Assignments

Printing the First Line

of every file in a directory

Since every one of our Python programs started with a uniform comment line, listing the first line in every file could be the start of building an index.

```
# L-23 MCS 260 Friday 19 October 2007: firstline
#
# This program prints the first line in every
# file in the current directory.
#
import os
L = os.listdir('.')
for filename in L:
    print 'first line in ' + filename + ':'
    file = open(filename,'r')
    s = file.readline()
    print s[:-1]    # leave newline off
    file.close()
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Printing the First Line

of every file in a directory

Since every one of our Python programs started with a uniform comment line, listing the first line in every file could be the start of building an index.

```
# L-23 MCS 260 Friday 19 October 2007: firstline
#
# This program prints the first line in every
# file in the current directory.
#
import os
L = os.listdir('.')
for filename in L:
    print 'first line in ' + filename + ':'
    file = open(filename,'r')
    s = file.readline()
    print s[:-1]    # leave newline off
    file.close()
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Printing the First Line

of every file in a directory

Since every one of our Python programs started with a uniform comment line, listing the first line in every file could be the start of building an index.

```
# L-23 MCS 260 Friday 19 October 2007: firstline
#
# This program prints the first line in every
# file in the current directory.
#
import os
L = os.listdir('.')
for filename in L:
    print 'first line in ' + filename + ':'
    file = open(filename,'r')
    s = file.readline()
    print s[:-1]    # leave newline off
    file.close()
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Printing the First Line

of every file in a directory

Since every one of our Python programs started with a uniform comment line, listing the first line in every file could be the start of building an index.

```
# L-23 MCS 260 Friday 19 October 2007: firstline
#
# This program prints the first line in every
# file in the current directory.
#
import os
L = os.listdir('.')
for filename in L:
    print 'first line in ' + filename + ':'
    file = open(filename,'r')
    s = file.readline()
    print s[:-1]    # leave newline off
    file.close()
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

File Methods

rename and remove

method	description
<code>rename ()</code>	takes two names as arguments to rename a file or directory from the first into the second name
<code>remove ()</code>	removes the file with name given as argument

Application: remove all Python byte compiled files from the current directory (files that end in `.pyc`).

Algorithm:

1. get a listing of all files;
2. check all files for their extension;
3. delete those files that end in `.pyc`.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

File Methods

rename and remove

method	description
<code>rename ()</code>	takes two names as arguments to rename a file or directory from the first into the second name
<code>remove ()</code>	removes the file with name given as argument

Application: remove all Python byte compiled files from the current directory (files that end in `.pyc`).

Algorithm:

1. get a listing of all files;
2. check all files for their extension;
3. delete those files that end in `.pyc`.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

File Methods

rename and remove

method	description
<code>rename ()</code>	takes two names as arguments to rename a file or directory from the first into the second name
<code>remove ()</code>	removes the file with name given as argument

Application: remove all Python byte compiled files from the current directory (files that end in `.pyc`).

Algorithm:

1. get a listing of all files;
2. check all files for their extension;
3. delete those files that end in `.pyc`.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

File Methods

rename and remove

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>rename ()</code>	takes two names as arguments to rename a file or directory from the first into the second name
<code>remove ()</code>	removes the file with name given as argument

Application: remove all Python byte compiled files from the current directory (files that end in `.pyc`).

Algorithm:

1. get a listing of all files;
2. check all files for their extension;
3. delete those files that end in `.pyc`.

File Methods

rename and remove

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>rename ()</code>	takes two names as arguments to rename a file or directory from the first into the second name
<code>remove ()</code>	removes the file with name given as argument

Application: remove all Python byte compiled files from the current directory (files that end in `.pyc`).

Algorithm:

1. get a listing of all files;
2. check all files for their extension;
3. delete those files that end in `.pyc`.

File Methods

rename and remove

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>rename ()</code>	takes two names as arguments to rename a file or directory from the first into the second name
<code>remove ()</code>	removes the file with name given as argument

Application: remove all Python byte compiled files from the current directory (files that end in `.pyc`).

Algorithm:

1. get a listing of all files;
2. check all files for their extension;
3. delete those files that end in `.pyc`.

Removing .pyc Files

the program delpyc.py

```
# L-23 MCS 260 Fri 19 Oct 2007 : delpyc.py
#
# This program deletes all files in the
# current directory that end with .pyc.
#
import os
L = os.listdir('.')
for filename in L:
    if filename[-3:] == 'pyc':
        print 'deleting ' + filename
        os.remove(filename)
    else:
        print 'keeping ' + filename
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Removing .pyc Files

the program `delpyc.py`

```
# L-23 MCS 260 Fri 19 Oct 2007 : delpyc.py
#
# This program deletes all files in the
# current directory that end with .pyc.
#
import os
L = os.listdir('.')
for filename in L:
    if filename[-3:] == 'pyc':
        print 'deleting ' + filename
        os.remove(filename)
    else:
        print 'keeping ' + filename
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Removing .pyc Files

the program `delpyc.py`

```
# L-23 MCS 260 Fri 19 Oct 2007 : delpyc.py
#
# This program deletes all files in the
# current directory that end with .pyc.
#
import os
L = os.listdir('.')
for filename in L:
    if filename[-3:] == 'pyc':
        print 'deleting ' + filename
        os.remove(filename)
    else:
        print 'keeping ' + filename
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Removing .pyc Files

the program `delpyc.py`

```
# L-23 MCS 260 Fri 19 Oct 2007 : delpyc.py
#
# This program deletes all files in the
# current directory that end with .pyc.
#
import os
L = os.listdir('.')
for filename in L:
    if filename[-3:] == 'pyc':
        print 'deleting ' + filename
        os.remove(filename)
    else:
        print 'keeping ' + filename
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Directory Methods

an overview

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>listdir()</code>	takes as argument a directory name and returns a list of file names in it
<code>getcwd()</code>	requires no arguments and returns the name of the current working directory
<code>chdir()</code>	takes as argument a name of a directory and changes the current directory to that
<code>mkdir()</code>	takes as argument a name for a new directory creates a new directory in the current directory
<code>rmdir()</code>	takes as argument a name of an empty directory and deletes that directory

Applied in programs

1. that produce many different outputs;
2. to store intermediate results temporarily.

Directory Methods

an overview

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>listdir()</code>	takes as argument a directory name and returns a list of file names in it
<code>getcwd()</code>	requires no arguments and returns the name of the current working directory
<code>chdir()</code>	takes as argument a name of a directory and changes the current directory to that
<code>mkdir()</code>	takes as argument a name for a new directory creates a new directory in the current directory
<code>rmdir()</code>	takes as argument a name of an empty directory and deletes that directory

Applied in programs

1. that produce many different outputs;
2. to store intermediate results temporarily.

Directory Methods

an overview

method	description
<code>listdir()</code>	takes as argument a directory name and returns a list of file names in it
<code>getcwd()</code>	requires no arguments and returns the name of the current working directory
<code>chdir()</code>	takes as argument a name of a directory and changes the current directory to that
<code>mkdir()</code>	takes as argument a name for a new directory creates a new directory in the current directory
<code>rmdir()</code>	takes as argument a name of an empty directory and deletes that directory

Applied in programs

1. that produce many different outputs;
2. to store intermediate results temporarily.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Directory Methods

an overview

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>listdir()</code>	takes as argument a directory name and returns a list of file names in it
<code>getcwd()</code>	requires no arguments and returns the name of the current working directory
<code>chdir()</code>	takes as argument a name of a directory and changes the current directory to that
<code>mkdir()</code>	takes as argument a name for a new directory creates a new directory in the current directory
<code>rmdir()</code>	takes as argument a name of an empty directory and deletes that directory

Applied in programs

1. that produce many different outputs;
2. to store intermediate results temporarily.

Directory Methods

an overview

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>listdir()</code>	takes as argument a directory name and returns a list of file names in it
<code>getcwd()</code>	requires no arguments and returns the name of the current working directory
<code>chdir()</code>	takes as argument a name of a directory and changes the current directory to that
<code>mkdir()</code>	takes as argument a name for a new directory creates a new directory in the current directory
<code>rmdir()</code>	takes as argument a name of an empty directory and deletes that directory

Applied in programs

1. that produce many different outputs;
2. to store intermediate results temporarily.

Directory Methods

an overview

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

method	description
<code>listdir()</code>	takes as argument a directory name and returns a list of file names in it
<code>getcwd()</code>	requires no arguments and returns the name of the current working directory
<code>chdir()</code>	takes as argument a name of a directory and changes the current directory to that
<code>mkdir()</code>	takes as argument a name for a new directory creates a new directory in the current directory
<code>rmdir()</code>	takes as argument a name of an empty directory and deletes that directory

Applied in programs

1. that produce many different outputs;
2. to store intermediate results temporarily.

The Module `os.path`

MCS 260 L-23

19 October 2007

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

If a name is mistyped, our program should not crash.

If the user provides `name` for a file, then `os.path.exists(name)` will return `True` or `False` depending on whether the file exists or not.

To check whether `name` is the name of a directory, use `os.path.isdir(name)` or `os.path.isfile(name)` to see if `name` is a file.

Both methods return `True` or `False`

The Module `os.path`

MCS 260 L-23

19 October 2007

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

If a name is mistyped, our program should not crash.

If the user provides `name` for a file, then `os.path.exists(name)` will return `True` or `False` depending on whether the file exists or not.

To check whether `name` is the name of a directory, use `os.path.isdir(name)` or `os.path.isfile(name)` to see if `name` is a file.

Both methods return `True` or `False`

The Module `os.path`

MCS 260 L-23

19 October 2007

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

If a name is mistyped, our program should not crash.

If the user provides `name` for a file, then `os.path.exists(name)` will return `True` or `False` depending on whether the file exists or not.

To check whether `name` is the name of a directory, use `os.path.isdir(name)` or `os.path.isfile(name)` to see if `name` is a file.

Both methods return `True` or `False`

Information Methods in os.path

Three methods to retrieve information, all take a the name of a file as argument:

method	description
<code>os.path.getsize()</code>	returns size of the file
<code>os.path.getatime()</code>	returns time when last accessed
<code>os.path.getmtime()</code>	returns time when last modified

Application: print the name of the largest file in the current directory.

Algorithm:

1. **initialize** the name of the largest file and its size using the name and size of the first file
2. compare the size of all other files to the maximum
3. adjust the name and size of the current largest file whenever a larger file is found

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

[Summary +
Assignments](#)

Information Methods in os.path

Three methods to retrieve information, all take a the name of a file as argument:

method	description
<code>os.path.getsize()</code>	returns size of the file
<code>os.path.getatime()</code>	returns time when last accessed
<code>os.path.getmtime()</code>	returns time when last modified

Application: print the name of the largest file in the current directory.

Algorithm:

1. **initialize** the name of the largest file and its size using the name and size of the first file
2. compare the size of all other files to the maximum
3. adjust the name and size of the current largest file whenever a larger file is found

Information Methods in os.path

Three methods to retrieve information, all take a the name of a file as argument:

method	description
<code>os.path.getsize()</code>	returns size of the file
<code>os.path.getatime()</code>	returns time when last accessed
<code>os.path.getmtime()</code>	returns time when last modified

Application: print the name of the largest file in the current directory.

Algorithm:

1. **initialize** the name of the largest file and its size using the name and size of the first file
2. compare the size of all other files to the maximum
3. adjust the name and size of the current largest file whenever a larger file is found

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Information Methods in os.path

Three methods to retrieve information, all take a the name of a file as argument:

method	description
<code>os.path.getsize()</code>	returns size of the file
<code>os.path.getatime()</code>	returns time when last accessed
<code>os.path.getmtime()</code>	returns time when last modified

Application: print the name of the largest file in the current directory.

Algorithm:

1. **initialize** the name of the largest file and its size using the name and size of the first file
2. compare the size of all other files to the maximum
3. adjust the name and size of the current largest file whenever a larger file is found

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Information Methods in os.path

Three methods to retrieve information, all take a the name of a file as argument:

method	description
<code>os.path.getsize()</code>	returns size of the file
<code>os.path.getatime()</code>	returns time when last accessed
<code>os.path.getmtime()</code>	returns time when last modified

Application: print the name of the largest file in the current directory.

Algorithm:

1. **initialize** the name of the largest file and its size using the name and size of the first file
2. compare the size of all other files to the maximum
3. adjust the name and size of the current largest file whenever a larger file is found

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Finding the Largest File

the program maxfile.py

```
# L-23 MCS 260 Fri 19 Oct 2007 : maxfile.py
#
# This program prints the name of the largest
# file in the current directory.
#
import os.path
L = os.listdir('.')
if len(L) == 0:
    print 'empty directory'
else:
    max = os.path.getsize(L[0])
    name = L[0]
    for k in range(1,len(L)):
        s = os.path.getsize(L[k])
        if s > max:
            max = s
            name = L[k]
    print 'largest file ' + name \
        + ' has ' + str(max) + ' bytes'
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Finding the Largest File

the program maxfile.py

```
# L-23 MCS 260 Fri 19 Oct 2007 : maxfile.py
#
# This program prints the name of the largest
# file in the current directory.
#
import os.path
L = os.listdir('.')
if len(L) == 0:
    print 'empty directory'
else:
    max = os.path.getsize(L[0])
    name = L[0]
    for k in range(1,len(L)):
        s = os.path.getsize(L[k])
        if s > max:
            max = s
            name = L[k]
    print 'largest file ' + name \
    + ' has ' + str(max) + ' bytes'
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Finding the Largest File

the program maxfile.py

```
# L-23 MCS 260 Fri 19 Oct 2007 : maxfile.py
#
# This program prints the name of the largest
# file in the current directory.
#
import os.path
L = os.listdir('.')
if len(L) == 0:
    print 'empty directory'
else:
    max = os.path.getsize(L[0])
    name = L[0]
    for k in range(1,len(L)):
        s = os.path.getsize(L[k])
        if s > max:
            max = s
            name = L[k]
print 'largest file ' + name \
+ ' has ' + str(max) + ' bytes'
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Finding the Largest File

the program maxfile.py

```
# L-23 MCS 260 Fri 19 Oct 2007 : maxfile.py
#
# This program prints the name of the largest
# file in the current directory.
#
import os.path
L = os.listdir('.')
if len(L) == 0:
    print 'empty directory'
else:
    max = os.path.getsize(L[0])
    name = L[0]
    for k in range(1,len(L)):
        s = os.path.getsize(L[k])
        if s > max:
            max = s
            name = L[k]
print 'largest file ' + name \
+ ' has ' + str(max) + ' bytes'
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments

Finding the Largest File

the program `maxfile.py`

```
# L-23 MCS 260 Fri 19 Oct 2007 : maxfile.py
#
# This program prints the name of the largest
# file in the current directory.
#
import os.path
L = os.listdir('.')
if len(L) == 0:
    print 'empty directory'
else:
    max = os.path.getsize(L[0])
    name = L[0]
    for k in range(1,len(L)):
        s = os.path.getsize(L[k])
        if s > max:
            max = s
            name = L[k]
    print 'largest file ' + name \
        + ' has ' + str(max) + ' bytes'
```

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the `os` module
directory methods

Summary + Assignments

Summary + Assignments

MCS 260 L-23

19 October 2007

We ended chapter 7 in *Making Use of Python*;
for Unix, see §15.6.2 in *The Art & Craft of Computing*.

Assignments:

1. Write a Python program to count the number of Python programs (i.e.: files with extension ".py") in the current directory.
2. Write a Python program to classify the files in the current directory according to their extension, counting files ending in `.py`, `.txt`, `.exe`, etc.
3. Write a Python program to find the oldest file in the current directory.

Homework collected on Monday 22 October:

exercises 1 and 3 of Lecture 20; exercises 3 and 5 of Lecture 21; and exercise 2 of Lecture 22.

Files

archiving files with tar
concatenation, redirection
and pipes

Python

standard input and output
the os module
directory methods

Summary + Assignments