Input and Output

Mark Neal

Simple keyboard input

- Use `read` to enter simple terms and bind them to a variable:

```
read(X).
|: fish.
X = fish.
```

- Must terminate input with a “.”
- Can enter numbers, lists etc...
- Use standard Prolog syntax

Simple output

- Use write to display variables, lists, strings etc...

```
write(fish).
fish


read(X), write(X).


write('hello world').


write('please enter two lists to have them stuck together'), nl,
read(X), read(Y), length(X,L), nl, write(L), nl, append(X,Y,R),
write(R),nl.
```

- Remember prolog outputs any bindings it has made anyway, but if you use prolog in “script” mode this goes away `(swipl -s myfile.pl)`
- But in this mode you need to make your “query” into a rule:

```
. run :- read(X), write(x).
```

Simple file IO

- You can use `tell` and `told` to switch output to a file and back again

```
tell('wibble.txt').
```

- Redirects output to the file `wibble.txt` until

```
told.
```

- Sets it back to normal
- If you want to read things back you can use

```
see('wibble.txt').
```

- To use the file as the source of input, and

```
seen.
```

- To set it back to normal

Simple file IO

- But...
  - If you write things to a file like:

    `write([a, b, c]).`
  - And then try to read them back with:

    `read(A).`
  - It will fail :-(
  - Because when you read them back you need a dot
  - So use:

    `write([a, b, c]), write('.').`
  - Also re-opening will overwrite the file
  - When you reach the end of the file you get `end_of_file` from `read`