CLIPS 6.0: An Expert System Tool

Tool	Complete Environment for developing expert systems, including integrated editor and debugger
Shell	Performs Inferences and reasoning, i.e. uses the created rule and fact system

Basic Elements of an Expert System Shell

Facts A list of facts and instances that describe the current status of the system

Knowledge Base The rules which act on the facts to produce results

Inference Engine Controls the execution of the rules

CLIPS will be used as a concrete illustration of an expert system having the above elements. Each item should become clear as one gets familiar with CLIPS itself.

Of Absolute Importance

Calling CLIPS:

```
sun> clips

         Fuzzy CLIPS (V6.02 05/15/94)
CLIPS>
```

I want out:

```
CLIPS> (exit)
```

Facts

A key ingredient in an expert system are the **Facts**:

```
CLIPS> (assert (duck))
(assert (duck))
<Fact-0>
```

With the assert command, the fact has been added to the fact list. Since this is the first, it is fact number zero.
Note the LISP structure of the input (with lots of parentheses).

What facts do I have

```
CLIPS> (facts)
f-0     (duck) CF 1.00
For a total of 1 fact
```

f-0 is the identifier

reset

```
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
For a total of 1 fact
CLIPS> (assert (duck))
<Fact-1>
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (duck) CF 1.00
For a total of 2 facts
```

Duplicate Facts?

```
CLIPS> (assert (duck))
FALSE
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (duck) CF 1.00
For a total of 2 facts.
```

FALSE is returned saying that you cannot enter the same fact twice (this can be overridden, however)

Adding More Facts

```
CLIPS> (assert (quack))
<Fact-2>
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (duck) CF 1.00
f-2     (quack) CF 1.00
For a total of 3 facts.
```

Command: clear

```
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
f-1     (duck) CF 1.00
f-2     (quack) CF 1.00
For a total of 3 facts.
CLIPS> (clear)
CLIPS> (facts)
(facts)
```

(clear) clears everything to the state when the program was started. It differs from reset in that not only facts are cleared, but also rules.

Fact Indices

```
CLIPS> (clear)
CLIPS> (assert (a) (b) (c))
<Fact-2>
CLIPS> (facts)
f-0     (a) CF 1.00
f-1     (b) CF 1.00
f-2     (c) CF 1.00
For a total of 3 facts.
CLIPS> (facts 0)
f-0     (a) CF 1.00
f-1     (b) CF 1.00
f-2     (c) CF 1.00
For a total of 3 facts.
CLIPS> (facts 1)
(facts 1)
f-1     (b) CF 1.00
f-2     (c) CF 1.00
For a total of 2 facts.
CLIPS> (facts 2)
(facts 2)
f-2     (c) CF 1.00
For a total of 1 fact.
CLIPS> (facts 0 2 2)
(facts 0 2 2)
f-0     (a) CF 1.00
f-1     (b) CF 1.00
For a total of 2 facts.
```

Multiple Field Facts

Single Field:

```
(duck)
```

Two Fields: Hunter Brian shoots Duck

```
(Brian duck)
```

Ordering Significant: Duck Shoots Brian

```
(duck Brian)
```

Nothing is Significant: Duck didn't shoot anything

```
(duck nil)
```

The CLIPS system interprets this purely on a syntactical level, the actual meaning is left to the user and/or rules

A bit of Software Engineering

A better rule would be:

```
(hunter-game duck Brian)
```

The duck is the hunter and Brian is the game (oh dear)

Field Types

- float
- integer
- symbol
- string
- externaladdress
- factaddress
- instancename
- instanceaddress

The type of the field is determined by the type of the value stored in the field. In an unnamed field, the type is determined *implicitly* (more later)

Symbol Names

- Case Sensitive
- Delimintors:
  - any non-printable ASCII character: spaces, carriage returns, tabs and linefeeds
  - double quotes
  - opening and closing parentheses
  - ampersand
  - veritical bar
  - less than
  - tilde
  - semicolon

Examples

- duck
- duck1
- duck_soup
- duck-soup
- duck1-1_soup-soup

No Embedding of Facts

```
(duck (shot Brian Gary Rey))
```

Examples

```
CLIPS> (clear)
(clear)
CLIPS> (assert (number 1))
(assert (number 1))
<Fact-0>
CLIPS> (assert (x 1.5))
(assert (x 1.5))
<Fact-1>
CLIPS> (assert (y -1))
(assert (y -1))
<Fact-2>
CLIPS> (assert (z 65))
(assert (z 65))
<Fact-3>
CLIPS> (assert (distance 3.5e5))
(assert (distance 3.5e5))
<Fact-4>
CLIPS> (assert (coordinates 1 2 3))
(assert (coordinates 1 2 3))
<Fact-5>
CLIPS> (assert (coordinates 1 3 2))
(assert (coordinates 1 3 2))
<Fact-6>
```

```
CLIPS> (facts)
(facts)
f-0     (number 1) CF 1.00
f-1     (x 1.5) CF 1.00
f-2     (y -1) CF 1.00
f-3     (z 65) CF 1.00
f-4     (distance 350000.0) CF 1.00
f-5     (coordinates 1 2 3) CF 1.00
f-6     (coordinates 1 3 2) CF 1.00
For a total of 7 facts.
```

- Note the translation of the exponential.
- Also, note that a symbol is always required as the first field
- Facts can be ordered or unordered (coordinates)

Delimitors

```
CLIPS> (clear)
CLIPS> (assert (The duck says "Quack"))
<Fact-0>
CLIPS> (assert (The     duck      says      "Quack"))
FALSE
CLIPS> (assert (The
duck
says
"Quack"))
FALSE
CLIPS> (assert (The duck says "Quack
"))
<Fact-1>
CLIPS> (facts)
(facts)
f-0     (The duck says "Quack") CF 1.00
f-1     (The duck says "Quack
") CF 1.00
For a total of 2 facts.
```

Style

```
CLIPS> (assert (duck) (cow) (horse))
(assert (duck) (cow) (horse))
<Fact-2>
CLIPS> (facts)
(facts)
f-0     (duck) CF 1.00
f-1     (cow) CF 1.00
f-2     (horse) CF 1.00
For a total of 3 facts.
```

Does not show any relationship, but....

```
CLIPS> (assert (animal-is cow))
<Fact-0>
CLIPS> (assert (animal-is duck))
(assert (animal-is duck))
<Fact-1>
CLIPS> (assert (animal-is horse))
(assert (animal-is horse))
<Fact-2>
CLIPS> (facts)
(facts)
f-0     (animal-is cow) CF 1.00
f-1     (animal-is duck) CF 1.00
f-2     (animal-is horse) CF 1.00
For a total of 3 facts.
```

Relates them as being animals

Retraction of a Fact

```
CLIPS> (clear)
CLIPS> (assert (animal-is duck))
<Fact-0>
CLIPS> (assert (animal-sound quack))
<Fact-1>
CLIPS> (assert (The duck says "Quack"))
<Fact-2>
CLIPS> (facts)
f-0     (animal-is duck) CF 1.00
f-1     (animal-sound quack) CF 1.00
f-2     (The duck says "Quack") CF 1.00
For a total of 3 facts.
CLIPS> (retract 2)
CLIPS> (facts)
f-0     (animal-is duck) CF 1.00
f-1     (animal-sound quack) CF 1.00
For a total of 2 facts.
CLIPS> (retract 2)
[PRNTUTIL1] Unable to find fact f-2.
CLIPS> (retract 1)
CLIPS> (facts)
f-0     (animal-is duck) CF 1.00
For a total of 1 fact.
CLIPS> (retract 0)
CLIPS> (facts)
```

```
CLIPS> (clear)
CLIPS> (assert (a) (b) (c))
<Fact-2>
CLIPS> (facts)
f-0     (a) CF 1.00
f-1     (b) CF 1.00
f-2     (c) CF 1.00
For a total of 3 facts.
CLIPS> (retract 0 2)
CLIPS> (assert (d) (e) (f))
<Fact-5>
CLIPS> (facts)
f-1     (b) CF 1.00
f-3     (d) CF 1.00
f-4     (e) CF 1.00
f-5     (f) CF 1.00
For a total of 4 facts.
CLIPS> (retract *)
CLIPS> (facts)
CLIPS>
```

Expert System Debugging Command: Watch

```
CLIPS> (clear)
CLIPS> (watch facts)
CLIPS> (assert (animal-is duck))
==> f-0     (animal-is duck) CF 1.00
<Fact-0>
CLIPS> (reset)
<== f-0     (animal-is duck) CF 1.00
==> f-0     (initial-fact) CF 1.00
CLIPS> (assert (animal-is duck))
==> f-1     (animal-is duck) CF 1.00
<Fact-1>
CLIPS> (retract 1)
<== f-1     (animal-is duck) CF 1.00
CLIPS> (facts)
f-0     (initial-fact) CF 1.00
For a total of 1 fact.
```