Title

stata.com

pragma — Suppressing warning messages

Description Syntax Remarks and examples Also see

Description

pragma informs the compiler of your intentions so that the compiler can avoid presenting misleading warning messages and so that the compiler can better optimize the code.

Syntax

```
pragma unset varname

pragma unused varname
```

Remarks and examples

stata.com

Remarks are presented under the following headings:

pragma unset

The pragma

```
pragma unset X
```

suppresses the warning message

```
note: variable X may be used before set.
```

The pragma has no effect on the resulting compiled code.

In general, the warning message flags logical errors in your program, such as

```
: function problem(real matrix a, real scalar j)
> {
>     real scalar i
>
>     j = i
>     ...
> }
note: variable i may be used before set.
```

Sometimes, however, the message is misleading:

```
: function notaproblem(real matrix a, real scalar j)
> {
>     real matrix V
>
>     st_view(V, ...)
>     ...
> }
note: variable V may be used before set.
```

In the above, function st_view() (see [M-5] st_view( )) defines V, but the compiler does not know that.

The warning message causes no problem but, if you wish to suppress it, change the code to read

```
: function notaproblem(real matrix a, real scalar j)
> {
>     real matrix V
>
>     pragma unset V
>     st_view(V, ...)
>     ...
> }
```

pragma unset V states that you know V is unset and that, for warning messages, the compiler should act as if V were set at this point in your code.

pragma unused

The pragma

```
pragma unused X
```

suppresses the warning messages

```
note: argument X unused.
note: variable X unused.
note: variable X set but not used.
```

The pragma has no effect on the resulting compiled code.

Intentionally unused variables most often arise with respect to function arguments. You code

```
: function resolve(A, B, C)
> {
>     ...
> }
note: argument C unused.
```

and you know well that you are not using C. You include the unnecessary argument because you are attempting to fit into a standard or you know that, later, you may wish to change the function to include C. To suppress the warning message, change the code to read

```
: function resolve(A, B, C)
> {
>     ...
>     pragma unused C
>     ...
> }
```

The pragma states that you know C is unused and, for the purposes of warning messages, the compiler should act as if *C* were used at this point in your code.

Unused variables can also arise, and in general, they should simply be removed,

```
: function resin(X, Y)
> {
>     real scalar i
>     ...
>     ... code in which i never appears
>     ...
> }
note: variable i unused.
```

Rather than using the pragma to suppress the message, you should remove the line `real scalar i`.

Warnings are also given for variables that are set and not used:

```
: function thwart(X, Y)
> {
>     real scalar i
>     ...
>     i = 1
>     ...
>     ... code in which i never appears
>     ...
> }
note: variable i set but unused.
```

Here you should remove both the `real scalar i` and `i = 1` lines.

It is possible, however, that the set-but-unused variable was intentional:

```
: function thwart(X, Y)
> {
>     real scalar i
>     ...
>     i = somefunction(...)
>     ...
>     ... code in which i never appears
>     ...
> }
note: variable i set but not used.
```

You assigned the value of `somefunction()` to `i` to prevent the result from being displayed. Here you could use `pragma unused i` to suppress the warning message, but a better alternative would be

```
: function thwart(X, Y)
> {
>     ...
>     (void) somefunction(...)
>     ...
> }
```

See *Assignment suppresses display, as does (void)* in [M-2] **exp**.

Also see

[M-2] **Intro** — Language definition