Introduction To Fortran Conversion

Nick Maclaren

nmm1@cam.ac.uk

July 2009

Overview of Course

A very brief history
Appropriate tools and techniques
New facilities not covered (and why!)

Why to change and what it will cost
Recognising and handling variants
Converting old constructions

How to convert to modern style
Take advantages of improvements

Beyond the Course

OldFortran/

http://www.nag.co.uk/sc22wg5/

http://www.fortran.com/fortran/
⇒ ‘Information’, ‘Standards Documents’

Please Note

Assumes experienced Fortran programmers
May assume too much, especially on old stuff

- Please interrupt if you don’t understand

It mentions constructs, doesn’t describe them
You will need to look them up for the details

- Or ask for help when converting your code

For more information on modern Fortran, see
Course “Introduction to Modern Fortran”

Fortran Timeline

FORTRAN – IBM Language 1956
FORTRAN II – IBM Language 1958
FORTRAN IV – IBM Language 1962
FORTRAN 66 – ANSI/ISO Standard 1972
FORTRAN 77 – ANSI/ISO Standard 1980

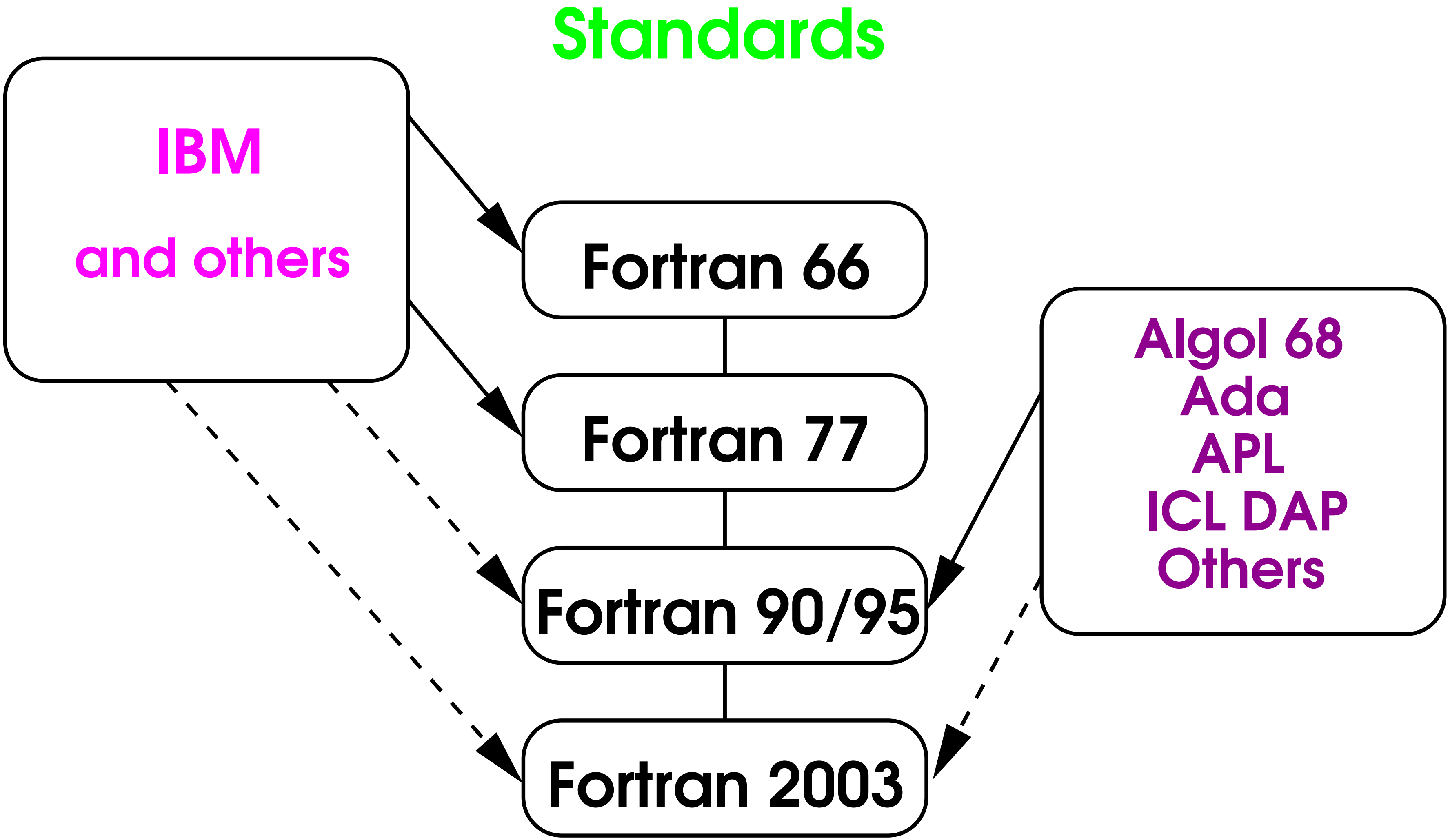
  FORTRAN IV faded away

Fortran 90 – ISO Standard 1991

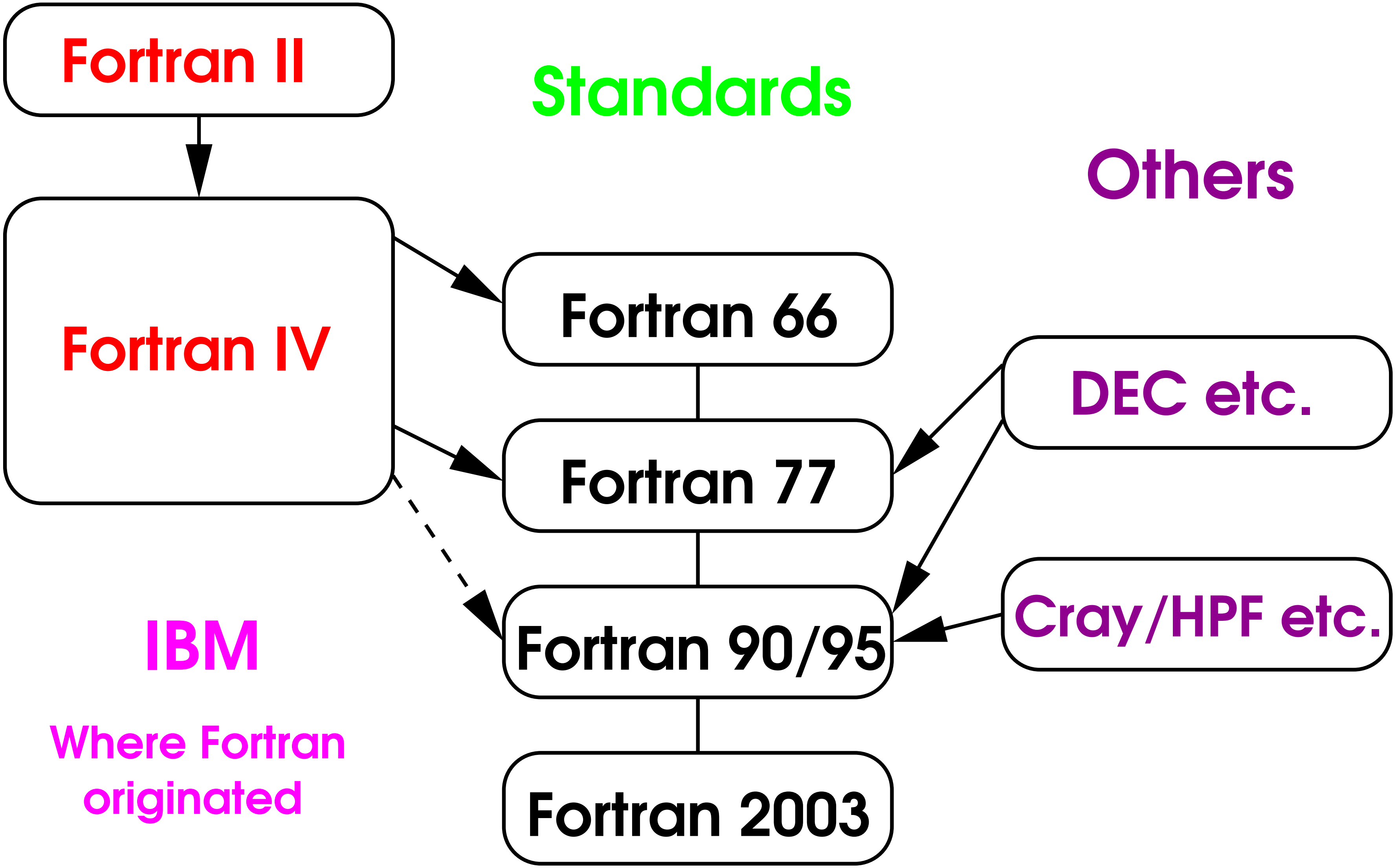
  The most extreme variants faded away

Fortran 95 – ISO Standard 1996
Fortran 2003 – ISO Standard 2004
Fortran 2008 – expected in 2010

Fortran Ancestry



Fortran Conversions



Aside: Source Format Selection

Despite belief, not in scope of the standard
Sometimes options –free, –fixed or similar

Another very common convention is:
fred.F, fred.f assumed in fixed–format
fred.F90, fred.f90 assumed in free–format
fred.F90, fred.F put through cpp first

Mistakes cause a flood of error messages
Included files had better be same format

Appropriate Tools

Fortran syntax may be messy, but it is clean
You don't need a compiler to handle it
Write a Python/Perl tool in an hour or two

http://www.fortran.com/f2f90.tar.gz
This isn't very clever – fix code first

There are also some good NAGWare tools
No longer marketed – but ask me if you need help

Good Coding Style

More modern, cleaner design is good style
Replacing superseded features is good style
Other style not within remit of this course
'Good style' also gets very religious

A few new features should be avoided
Some bad, old ones are unavoidable
Bad code can be written in any language

Aside: The F Language

A (true?) subset of Fortran 90
Implemented before Fortran 90, so popular
Some people believe it is better style

No conversion needed to run a subset!
Program in F if you like its style

- Ignore it completely if you don't

http://www.fortran.com/F/

Disclaimer

This won’t cover everything!
Still learning modern Fortran myself
Fortran has a lot of historical relics

- Many compilers are only Fortran 95

Fortran 2003 has improved several areas
It still has several futile restrictions

- Examples not given if advice is do nothing
- Examples are simple, NOT good practice

Converting Your Code

- Don't make changes without a reason

Almost all old Fortran is still legal
But a mixture of styles can be a problem

- No need to do everything at once

Generally, clean up code as you work on it
Or deal with one aspect, globally

Improving portability is always good
But any change can break working code!

General Principles

ALWAYS save copy of original source

If possible, test program and save output

- Convert one area or aspect completely
- Retest and compare output with previous
- Save scripts, source and results

You may need to repeat several stages back!

Want to be able to use diff if possible

Systematic Changes

Some aspects best done to whole program
Precision conversion is extreme example
Or fixed⇒free format conversion

- Use automatic tools if at all possible

By hand is very tedious and error–prone

- Now test, check and save state

Manual Changes

Some things can be done only manually
Code restructuring, rewriting of variants

Very error–prone, best done when rewriting
Consider whether you need to do them

Sometimes you can write a special tool
MUCH faster and more reliable
Python, Perl, awk, whatever you like

Example of Compromise

Make MODULE from COMMON manually
Including/rewriting any BLOCK DATA

- Find files that use it with grep
- Script to add USE & remove old declarations

Python, Perl, awk, grep/sed, . . .

Watch out for similar, unrelated declarations
grep/sed are rarely powerful enough

Totally New Features

Not covered directly in this course
Anything with no analogue in Fortran 77
Except where existing code emulates them

Mainly semantic extension etc.
Called object oriented programming
Can be very hard to add to old code
Add as you redesign parts of your code

- Nothing further is said about that aspect!

Safe New Features

Mostly covered in Modern Fortran course

Environment and arguments – Aha! At last!
Can remove some old system–dependent code

Unfortunately, this is only in Fortran 2003

Risky New Features

Potentially useful, but a minefield
More system issues than seems possible

IEEE 754 a.k.a. ISO/IEC 60559 exceptions
Course “How Computers Handle Numbers”

C interoperability – vaguely and in theory
Course “Mixed–Language Programming”

Asynchronous declarations and I/O
VOLATILE is toxic – ask offline why

NAMELIST I/O

This new feature is not recommended
It has no equivalent in other languages

NAMELIST was included in Fortran 90
Fortran IV feature not in Fortran 77
Has some arcane restrictions on its use

- Avoid this in new code, if possible
- But don't bother to remove from old code

Clean up such code only as you rewrite

Optional Features

Optional extensions to the standard:

- Varying length strings
- Conditional compilation (‘Coco’)

Not many compilers support either
There is open source code for both

Miscellaneous

Lots of other minor improvements
Mostly not worth a conversion campaign

Good idea to leaf through book on Fortran 90/95
Check if messy code can be cleaned up

Almost all Fortran 66/77/90/95 is Fortran 2003
Replace old code as and when you work on it

Fortran 2003 Incompatibilities

A few with each of Fortran 77, 90 and 95
They are mostly mind–bogglingly obscure
A few, minor, rarely–used, features deleted
A very little code arguably changes meaning

Less than normal system–dependent variations
Main ones are covered later in this course

- Most old code will still work, unchanged

Even some code from 40+ years back!

Fortran 90/95/2003 Extensions

Mostly to enable parallelism
Directives generally as comments
Can simply compile for serial use

Very often several extra intrinsic functions
Need to find or write serial versions

- Ask for help if want to run in parallel

Well beyond scope of this course

Fortran 2008 will add coarrays

OpenMP

!OMP directive
COMP directive
*OMP directive

Extra instrinsics with names OMP_*

- Can often be run minor serially just as it is

Many compilers provide a stub library
(Simple) example code in specification

http://www.openmp.org/

HPF

!HPF$ directive
CHPF$ directive
*HPF$ directive

EXTRINSIC statement
HPF_LIBRARY built–in module

- Effectively superseded by OpenMP

Convert to OpenMP and/or seek advice

http://hpff.rice.edu/

Older Parallel Extensions

IBM and Alliant were first commercial
Have been a lot since – most rarely seen

DEC (VAX VMS) CPAR$, CDEC$ directives
Hasn’t been supported for a decade
Probably some HP guides on conversion

Old Cray CDIR$ directives
Similar remarks to DEC’s VAX VMS ones

Co-Array Fortran

```
REAL, DIMENSION(20)[20,*] :: A
INTEGER :: IB[*]
A(5)[3,7] = IB(5)[3]
A(:)[2,3] = C[1]
```

Quite a few extra intrinsic functions

Favoured by New Cray/NASA/DoD/etc. people
Can be converted to OpenMP

http://www.co–array.org/

Fortran Standard Coarrays

Watch This Space

Sun Interval Arithmetic

INTERVAL :: a
a = [0.1,0.3]
Extra operators (e.g. .SP.) and functions

In theory, could be mapped to pure Fortran 90

- In practice, just use Sun's compilers

Free for Solaris and Linux SPARC & Intel/AMD

http://developers.sun.com/sunstudio/index.jsp

So What Do We Change?

Reminder: this is not a complete list
My personal view of how to upgrade

But most experts will agree with me
Directions are clear, details aren't

- For each change, balance gain vs pain

Each decision depends on your requirements
Remember you can select aspects to change

Ancient Fortran

- Things that need changing, now

Many are already hindering portability
Some will still work, sometimes . . .
Others are dead or almost totally dead

The slides are online – mostly for historical interest
If you hit those problems, please ask

Merely Old Fortran

- Things to take advantage of modern features

Mostly for ‘‘software engineering’’
Clarity, maintainability, error checking etc.
No old code will break in forseeable future

We shall now go over the points I cover