



Wilhelm Gehrke, Herbert Schmid

Formel-Meister

Fortran-95-Compiler für Linux und Windows

Wenn Wissenschaftler komplizierte Berechnungen oder komplexe Simulationen ausführen wollen, greifen sie am liebsten zu Supercomputern und einer Programmiersprache, die den dort gestellten Anforderungen gerecht wird: Fortran. Mit zunehmender Rechenleistung der Büro-Rechner und steigender Beliebtheit von Linux-Clustern erobert die Sprache immer mehr herkömmliche PCs. Doch welchen Compiler nimmt man, wenn höchste Leistung gefragt ist? Welcher verhilft zu portablen Programmen, die später auch auf dem Supercomputer der Uni laufen?

Egal ob Wetter, Biochemie oder Autokonstruktion, an der Programmiersprache Fortran führt kein Weg vorbei. So läuft am derzeit schnellsten Supercomputer Deutschlands, der Hitachi SR8000-F1/168 am Leibniz Rechenzentrum in München, zum Großteil Fortran-Code. (Der Rechner belegt übrigens Platz 14 in der aktuellen Top-500-Liste der Super-

computer vom Juni.) Irmgard Frank vom Department Chemie der Universität München simulierte damit zum Beispiel die Reaktion des Sehpurpurs (Rhodopsin) in der Netzhaut des Auges auf sichtbares Licht. Dieser Vorgang initiiert eine Kaskade weiterer Reaktionen, die letzten Endes zu einem Nervenreiz führen. Die Simulation dieses komplexen Prozesses ist

besonders rechenintensiv, da dabei die quantenmechanische Vielteilchentheorie zum Tragen kommt.

Aber nicht nur die akademische Wissenschaft an den Universitäten, sondern auch die Forscher in Privatunternehmen greifen zu Fortran. Die tägliche Wettervorhersage des Deutschen Wetterdienstes stammt etwa von einem Fortran-Programm, das den gesamten Vorhersagezyklus von der Analyse bis zur Prognose erledigt. Zur Abfrage der in einer Oracle-Datenbank gespeicherten Messwerte arbeitet es allerdings mit einem in C verfassten Programmteil zusammen. Für ihre Kunden in der Industrie betreibt Electronic Data Systems (EDS) weltweit mehrere größere Rechenzentren, darunter einen Supercomputer in Rüsselsheim für Opel, der hauptsächlich Crashtests simuliert oder die Luftströmung an einzelnen Karosserievarianten berechnet.

Fachsprache

Aufgrund der Eigenschaften von Fortran lassen sich damit mathematische Algorithmen leicht formulieren: Die Sprache kennt komplexe Zahlen von vornherein, und Datenfelder mit variabler Größe programmiert

man, ohne sich gleich mit Pointer und dergleichen herumschlagen zu müssen. Sogar die parallele Verarbeitung mehrerer Programmteile formuliert man direkt in der Sprache und muss sie nicht wie in C mühevoll über Funktionsaufrufe nachbilden. Zudem existiert eine große Zahl an Bibliotheken, die mathematisch exakter arbeiten als etwa jene für C. Anders als bei C++ ist es bei Fortran für die Compiler recht einfach, den übersetzten Code zu optimieren; so sind etwa die Zusammenhänge zwischen einzelnen Arbeitsschritten gut sichtbar.

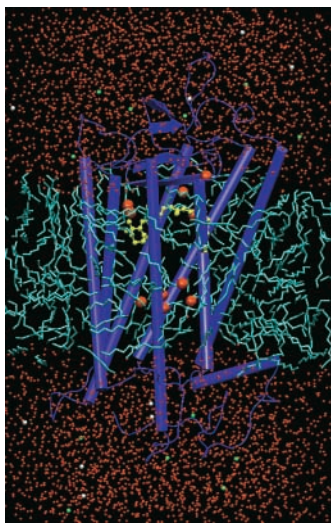
Doch schafft es nicht jeder Compiler gleich gut, die Möglichkeiten von Fortran optimal zu nutzen. Grund genug, neun Compiler für Windows und Linux genauer unter die Lupe zu nehmen. Dazu zogen wir die Fortran-Benchmarks des britischen Compiler-Händlers Polyhedron (www.polyhedron.co.uk) sowie die Benchmark-Suite CPU2000 der Standard Performance Evaluation Corporation (www.spec.org) heran. Beide liefen auf einem Athlon XP 2000+ (mit 1,66 GHz Takt) und einem Pentium 4 mit 2 GHz Takt. Häufig verwendeten Programme die Fortran-77- und Fortran-90-Bibliotheken von NAG oder IMSL. Wir haben zu jedem Compiler eine passende Variante gesucht und deren Zusammenarbeit überprüft.

Drei der getesteten Compiler sind sowohl in einer Variante für Linux als auch für Windows verfügbar. Bei Intel Fortran 6.0 und Absoft Pro Fortran 8.0 stehen beide auf dem gleichen Versionsstand, bei Fujitsu/Lahey hinkt die Windows-Variante (5.7 statt 6.1) ein wenig hinterher. Nur unter Linux nahm The Numerical Algorithms Group (NAG) teil, und das britische Softwarehaus Salford bietet seinen FTN95 nur für Windows an. Ebenfalls nur unter Windows läuft Compaq Visual Fortran, das heute nicht HP, sondern Intel gehört. Der Prozessorhersteller will es in den eigenen Compiler einschmelzen und das Ergebnis ab Frühjahr 2003 als Intel Fortran 7.0 vertreiben.

Compiler, die überwiegend auf massiv parallelen Systemen eingesetzt werden, kamen nicht in den Test, etwa das High Performance Fortran der Portland Group. Ebenfalls fehlen die bei-

den Fortran-77-Compiler GNU g77 und Salford FTN77, da sie den aktuellen Fortran-95-Standard noch nicht beherrschen. Sie liegen gratis als Download bereit (siehe Soft-Link).

Manche der genannten Compiler gibt es nicht nur für Windows oder Linux, sondern auch für weitere Plattformen. Intel und NAG unterstützen zum Beispiel noch die 64-bittige Itanium-Architektur, NAG hat zudem viele Unix-Varianten im Programm. Absoft liefert seine Compiler für Mac OS 9 oder X sowie Linux/PPC. Salfords Übersetzer erzeugt auch Binaries



So sieht der Mensch: Ein Fortran-Programm berechnete die Reaktion des Sehpurpurs (Rhodopsin) im Auge auf das einfallende Licht.

für .NET, wofür ansonsten nur noch Lahey eine Beta-Version anbietet. Seit kurzem hat Lahey auch eine Mac-OS-X-Fassung parat. AMD dürfte Absofts Absicht freuen, auch die 64-bittige Architektur der nächsten Jahr erwarteten Opteron-Prozessoren zu unterstützen.

Die Installation der meisten Compiler verlief problemlos: für Linux lieferten die Hersteller entweder eine rpm-Datei oder ein Tar-Archiv mit Install-Skript; die Windows-Versionen kamen mit einem Setup-Programm.

Unter Linux mussten wir das Einspielen der nötigen Lizenz-Dateien oft von Hand vornehmen. NAG verlangt selbst bei der Einzelplatzversion das aufwendige Aufsetzen eines eigenen Lizenz-Servers, ansonsten überraschte das Setup-Skript po-

sitiv: Es fragt sogar nach dem Kapitel, in das es die Man-Pages ablegen soll. Absofts Linux-Installation hingegen vergisst das Setzen von PATH und MAN-PATH in /etc/profile und erzeugt aber auch kein Shell-Skript als Ersatz. Die Manpages sind teilweise veraltet; ein Eintrag für die wichtige Umgebungsvariable 'ABSOFT_RT_FLAGS' fehlt etwa komplett.

Handarbeit

Intels Windows-Compiler benötigt ein wenig Vor- und Nacharbeit: Zuerst muss bereits eine englische Fassung von Microsofts Visual Studio 6.0 (inklusive Service Pack 5) installiert sein, da er auf dessen Runtime-Library zurückgreift. Das Programmierpaket gehört allerdings nicht zum Lieferumfang und man muss es getrennt erwerben. Die anschließende Installation auf einem deutschen Windows XP verursachte Probleme, da einige Komponenten fälschlicherweise in 'Program Files' statt in 'Programme' landeten. Kopiert man die Dateien ins richtige Verzeichnis, lässt sich das Add-in anschließend in Visual Studio anwählen.

Compaq Visual Fortran arbeitet mit einer ähnlichen Technik. Es bringt aber das Visual Studio (ohne C/C++-Compiler) gleich mit und ist daher vor solchen Problemen gefeit. Allerdings verlangt es nach der Installation von jedem Benutzer, ein zusätzliches 'Per User Setup' auszuführen.

Die Benchmarks von SPEC und Polyhedron zielen eigentlich auf reine Geschwindigkeitsmessungen und weniger auf die Qualität von Compilern. Doch die einzelnen Testprogramme liegen im Quelltext vor, die echten Anwendungen entstammen und daher 'gewachsenen' Code mitbringen. CPU2000 setzt einige Eigenheiten ein und enthält etwa Fortran-90-Programme im alten 'Lochkarten-Layout'. Beim Übersetzen sieht man daher, ob die Compiler mit fremden Fortran-Dialekten klarkommen.

Entwickelt man neue Programme, stört eine große Toleranz des Compilers, besonders wenn das Programm später auf dem Supercomputer der Uni laufen soll. Alle Compiler im Test halten für diesen Fall eine Option parat, die alle Erweiterungen

abschaltet und besonders strikt auf die Einhaltung des Fortran-95-Standards achtet.

Missverständnisse

Manche Probleme verursacht allerdings der an manchen Stellen zu locker gefasste Standard selbst. Die Genauigkeit von Variablen etwa legen KIND-Values fest, doch sind diese nicht weiter definiert. NAG und Salford zählen die Varianten von 1 bis 3 durch, alle anderen Hersteller verwenden die Zahl der benutzten Bytes (2, 4 oder 8), die auch die Benchmarks der SPEC einsetzen. Dem Compiler von NAG bringt man diese Definition einfach per Kommandozeilen-Option bei, und danach übersetzt er die Tests fehlerfrei.

Bei Salford hilft es, die Genauigkeit der Variablen mit einem Switch festzulegen, was bei manchen Benchmarks klappt. An einem der SPEC-Benchmarks (187.facerec) scheiterte der Compiler dann doch mit einem internen Fehler. Ärgerli-

cherweise zeigt er diesen in einer Message-Box an und blockiert damit unnötigerweise die Stapelverarbeitung mehrerer Quelltexte. Auch Laufzeitfehler von Kommandozeilenprogrammen werden per Message-Box angezeigt. Dadurch kann man Kompilate nicht über Nacht unbeaufsichtigt durchlaufen lassen.

Lahey ist der Fortran-Standard zu eng gefasst. Der Hersteller erweitert ihn kurzerhand um Eigenes und macht das dann auch noch zur Voreinstellung. Unglücklicherweise verwendet er dafür so häufige Namen wie 'Error' und damit scheitert der Compiler an den SPEC-Benchmarks 173.applu und 200.six-track. Schaltet man diese Erweiterungen mit -nli ab, gelingt die Übersetzung.

NAG hingegen sieht den Standard sehr streng, die Ein-/Ausgabe im alten Fortran-66-Stil etwa muss ausdrücklich erlaubt werden (-hollerith-io). Allerdings kann man mit -dusty und -mismatch_all einiges der Strenge zurücknehmen. Der SPEC-Benchmark

Lang gereift

Fortran entstand als einfacher *Formula Translator* bereits Mitte der 50er und leidet deshalb ein wenig unter einem angestaubten Image. Zu Unrecht, erweiterten doch laufend neue Standards die Sprache um aktuelle Programmierkonstrukte. Die erste genormte Fassung, Fortran 66, fasste primär die unterschiedlichsten Entwicklungen der Compilerhersteller zusammen. Allerdings wählte man dabei den Weg des geringsten Widerstands und entschied sich für eine vergleichsweise kleine Untermenge aller damals vorhandenen Erweiterungen. Fortran 77 brachte dann den Durchbruch. Damit hielt die strukturierte Programmierung – wenn auch noch mit Einschränkungen – Einzug in die Programmiersprache und räumte mit dem vorher herrschenden Spaghetti-Code auf.

Fortran 90 brach mit dem alten starren Spaltenlayout und führte ein alternatives 'Free Format' ein. Außerdem vergrößerte es die Möglichkeiten der strukturierten Programmierung mit neuen Kontrollstrukturen

und fügte benutzerdefinierte Datentypen sowie dynamische Felder und Zeiger hinzu. Fortran 95 erweiterte die Sprache nur geringfügig, vereinte aber das parallel entstandene High Performance Fortran (HPF) wieder mit dem regulären Standard.

Jeder neue Standard erlaubte es, von unbedeutenden Ausnahmen abgesehen, die mit dem Vorgänger verfassten Programme weiterhin zu übersetzen. Auch das für 2004 angekündigte Fortran 2000 soll diese Politik beibehalten; die Sprache unterstützt dann UNICODE-Zeichensätze und viele zusätzliche Sprachmittel, etwa für objektorientierte Techniken.

Ein Binärformat, etwa für Moduldateien oder Bibliotheken, schreibt allerdings kein Standard fest. Bibliotheken sollte man daher immer als ausdrücklich für den eingesetzten Compiler geeignet erwerben; für die Zusammenarbeit mit C enthalten manche Fortran-Compiler eigene Kompatibilitäts-Flags.

Benchmark-Ergebnisse auf AMD Athlon XP 2000+

Compiler	Kompilierzeit « besser ¹	Capacita « besser	Channel « besser	Fatigue « besser	SPEC(Fortran) besser ²
Absoft Pro Fortran 8.0 (Linux)	80,3	224	60	30	447
Intel Fortran Compiler 6.0 (Linux)	77,2	203	63	44	536
Fujitsu/Lahey Fortran 95 (Linux)	13,1	221	64	41	430
NAG f95 2.4 Edit 500 (Linux)	21,9	257	63	41	436
Absoft Pro Fortran 8.0 (Windows)	94,2	220	61	29	445
Cpq Visual Fortran 6.6A (Windows)	44,7	224	68	20	496
Intel Fortran Compiler 6.0 (Windows)	56,5	188	61	47	531
Fujitsu/Lahey Fortran 95 (Windows)	11,6	219	60	45	407
Salford FTN95 3.0 (Windows)	4,3	152	103	56	- ³

¹ aller 10 Fortran-Programme des Polyhedron Fortran-90-Benchmarks
² geometrisches Mittel der Ergebnisse von 168.wupwise, 171.swim, 172.mgrid, 173.applu, 178.galgel, 187.facerec, 189.lucas, 191.fma3d, 200.sixtrack, 301.apsi; nicht mit SPECfp-Werten vergleichbar
³ lief nicht fehlerfrei

191.fma3d enthält einen knapp 100 Zeilen fassenden Befehl; auf solch viele *Continuation Lines* bereit den NAG-Compiler die Option `-maxcontin` vor.

Mit einem quasi historisch bedingten Problem muss man sich unter Windows herumschlagen. Fortran-Programme auf Großrechnern nützen das erste Zeichen jeder Zeile, um die Ausgabe auf Formulare zu steuern (*Carriage Control Character*). Unter Unix ist das eher unüblich und die Programmierer – auch die der SPEC – verwenden das erste Zeichen wie jedes andere. Die Windows-Compiler von Lahey und Salford versuchen, die Formular-Steuerung nachzubilden und verschlucken das Zeichen dabei. Lahey löst das Problem mit dem Compiler-Switch `-lfe "Nnoctpr"`, vergisst diesen aber zu dokumentieren. Damit läuft die CPU2000 problemlos. Der Support von Salford konnte uns keinen derartigen Schalter nennen, weshalb deren Compiler neben 187.facerec an vier weiteren SPEC-Benchmarks scheiterte.

Klassiker

Geradezu typisch für Fortran-Programme ist der immer wieder zu kleine Stack, was besonders die Benchmarks von Polyhedron

aufzeigten. Erfreulicherweise fanden aber gleich mehrere Compiler (Absoft, Intel, Lahey und NAG) die passende Stack-Größe für jeden Benchmark. Alle anderen kennen eine Option, um die Größe genügend hoch zu setzen. Salford verlangt umständlich einen getrennten Aufruf von Slink. Unter Linux wachen zusätzlich eigene Limits des Betriebssystems über die Größe von Daten- und Stacksegment. Am besten hebt man sie mit `ulimit -d unlimited -s unlimited` auf.

Setzt ein Programm eine der IMSL- oder NAG-Bibliotheken ein, muss man bei der Wahl der Compiler aufpassen. Letztlich funktioniert nur Compaq Visual Fortran jeweils mit den Fortran-77- und den Fortran-90-Bibliotheken. Bei den anderen Herstellern muss man sich für einen der beiden Dritthersteller entscheiden oder auf die aktuellen Fortran-90-Fassungen verzichten (siehe Tabelle). Gibt es die benötigte Bibliothek nicht für den gewünschten Compiler, bleibt als Ausweg nur, die alten Programme auf die neue Umgebung zu portieren.

Oft schwirrt auch noch alter Fortran-77-Code herum. Daher mussten alle Compiler auch die Fortran-77-Benchmarks von Polyhedron absolvieren, was auch

fast allen gelang. Nur die Compiler von NAG und Salford erzeugten fehlerhafte Programme, die zur Laufzeit abstürzten.

Sprinter

Die Performance-Krone gebührt eindeutig Intels Compiler. Besonders deutlich tritt der Vorsprung auf Pentium-4-Rechnern hervor, unter Linux holt er bei der SPEC knappe 50 Prozent mehr heraus als der zweitplatzierte Compiler von Absoft. Bei einzelnen Tests muss sich Intels Compiler zwar knapp geschlagen geben, unterm Strich hält er seinen großen Abstand.

Intels gutes Abschneiden beruht aber auch darauf, dass alle anderen Compiler nur für den Pentium III oder noch ältere Prozessoren optimieren. Es entstehen Programme, mit denen der Athlon XP besonders gut und der Pentium 4 nun mal ziemlich schlecht umgehen kann. Aus demselben Grund erzielen alle außer Intel auf dem deutlich langsamer getakteten Athlon XP bessere Ergebnisse. Dahinter steckt keine böse Absicht – Intel schafft es halt als einziger, die SSE2-Einheit des hauseigenen Flaggschiffes zu nutzen.

Gelegentlich gehen die Optimierer aber zu weit und verbes-

seren das Programm zu Tode: Es stürzt ab, bleibt hängen oder verrechnet sich. Absofts Linux-Compiler erzeugt manchmal Code, der auf Pentium-4-Systemen falsch zählt. Der Support kommentierte das knapp mit 'You've found a bug in our compiler.' und verhalf uns mit der undokumentierten Option `+B138` zu lauffähigen Programmen.

Herzinfarkt

NAG antwortete auf einen von uns entdeckten Fehler einfach mit einer Zwischenrelease (Edit 500), die dann alles fehlerfrei übersetzte.

Intels Fortran kann als Einziges Programme in zwei Durchgängen übersetzen, was oft besonders gute Ergebnisse bringt. Die Kompilate des ersten Laufs erzeugen beim Rechnen Profildaten, beim zweiten nutzt der Compiler die gewonnenen Informationen, um das eigentliche Programm gezielt zu optimieren (Feed Back Optimization). Bei der SPEC verursacht der Profiler allerdings Probleme unter Linux. Nutzt man die höchste Optimierungsstufe für Pentium-III-Prozessoren, bleibt der SPEC-Benchmark '178.galgel' hängen.

Benchmark-Ergebnisse auf Intel Pentium 4 2.0 GHz

Compiler	Kompilierzeit « besser ¹	Capacita « besser	Channel « besser	Fatigue « besser	SPEC(Fortran) besser ²
Absoft Pro Fortran 8.0 (Linux)	66,5	228	43	40	425
Intel Fortran Compiler 6.0 (Linux)	79,1	181	51	15	622
Fujitsu/Lahey Fortran 95 (Linux)	12,6	232	51	51	388
NAG f95 2.4 Edit 500 (Linux)	23,0	254	41	76	379
Absoft Pro Fortran 8.0 (Windows)	96,0	242	39	42	436
Cpq Visual Fortran 6.6A (Windows)	42,6	241	39	31	478
Intel Fortran Compiler 6.0 (Windows)	56,0	182	39	58	559
Fujitsu/Lahey Fortran 95 (Windows)	12,5	237	42	70	356
Salford FTN95 3.0 (Windows)	3,9	191	69	79	- ³

¹ aller 10 Fortran-Programme des Polyhedron Fortran-90-Benchmarks
² geometrisches Mittel der Ergebnisse von 168.wupwise, 171.swim, 172.mgrid, 173.applu, 178.galgel, 187.facerec, 189.lucas, 191.fma3d, 200.sixtrack, 301.apsi; nicht mit SPECfp-Werten vergleichbar
³ lief nicht fehlerfrei

Für die hier veröffentlichten Messungen verzichten wir auf den Profiler, der in der Praxis ohnehin nur schwer zu verwenden ist: Er erfordert einen Trainingsatz mit Daten, die im Kleinen einen ähnlichen Programmablauf verursachen wie der spätere Einsatz. Als Ersatz setzen wir `-pod` beziehungsweise `-Qpod` ein. Diese Option verändert das Speicherlayout von Variablen und Arrays. Dabei ist eigentlich Vorsicht angebracht, so manches Fortran-Programm reagiert empfindlich auf derartige Eingriffe; es entstand aber bei keinem der Benchmarks ein Problem.

Auf besser optimierte Kompilate muss man jedenfalls länger warten. Absoft, Compaq und Intel benötigen zum Übersetzen der Polyhedron-Benchmarks mit 60 bis 100 Sekunden deutlich länger als die restlichen Compiler, die allesamt in

weniger als 25 Sekunden fertig wurden.

Fazit

Egal ob unter Linux oder Windows: Wenn es um möglichst schnelle Programme geht, ist Intels Compiler die ideale Wahl. Doch sobald es ein bisschen mehr sein soll, wird die Wahl schwieriger. Muss der Compiler etwa mit den Fortran-90-Bibliotheken von NAG und IMSL klarkommen und dann auch noch alten Digital Fortran Code verstehen, kommt Compaq Visual Fortran in den Blick. Doch wird es nicht mehr weiter entwickelt und man muss über kurz oder lang doch noch zu Intel wechseln.

Absofts Compiler verhält sich auf den unterschiedlichsten Plattformen nahezu identisch. Das erspart langwierige Portierungsarbeit in gemischten Um-

gebungen aus Windows- und Linux-Rechnern, unter die sich dank eigener Mac-OS- und Mac-OS-X-Versionen auch mal ein Mac mischen darf.

Wer Fortran lernen möchte oder hauptsächlich nur Code schreiben muss, der profitiert stark von den strengen Warnungen und Fehlermeldungen des NAG-Compilers. Gilt es gelegentlichen Abstürzen nachzuspüren, empfiehlt sich der Einsatz von Lahey Fortran. Dessen Runtime-Umgebung findet die meisten Laufzeitfehler, wie Polyhedron mit einem eigenen Test für Diagnose-Meldungen herausfand (www.polyhedron.co.uk/complnx.html).

Manche Entwickler brauchen ihren Quelltext im Blick und den Debugger immer gleich zur Hand. Mit Microsofts Visual Studio arbeiten Compaq Visual Fortran und Intels Windows-Compiler mit einer bekannten

und ausgereiften IDE. Diese liegt aber bei Intels Fortran nicht mit im Paket. Damit es funktioniert, muss man Microsofts Visual C++ zusätzlich erwerben.

Außerdem fiel die Entwicklungsumgebung von Salford positiv auf, da sie ohne Einarbeitung sofort verwendbar ist – nicht überfrachtet, aber trotzdem vollständig. Wegen zahlreicher interner Fehler und Abstürze eignet sich Salfords Compiler aber nur eingeschränkt für den alltäglichen Einsatz. Als Besonderheit erstellt er auch Programme für Microsofts .NET, und man kann damit erste Erfahrungen mit der neuen Umgebung sammeln. Aber dort erwächst Salford bereits Konkurrenz: Lahey hat ebenfalls schon eine Beta-Version für .NET. (hes)



Fortran-95-Compiler – Checkliste						
Compiler	Absoft Pro Fortran	Compaq Visual Fortran	Intel Fortran	Lahey Fortran Pro	NAG	Salford
	www.absoft.com	www.compaq.com/fortran	www.intel.com/developer	www.lahey.com	www.nag.co.uk	www.salford.co.uk
Vertrieb	QT Software www.qtsoftware.de	h.o.-COMPUTER www.hocomputer.de	h.o.-COMPUTER www.hocomputer.de	QT Software www.qtsoftware.de	LG Soft A S www.leipzigergewerbe.de/lgsoftas/	QT Software www.qtsoftware.de
	089 / 33 29 70	02 21 / 76 20 86	02 21 / 76 20 86	089 / 33 29 70	03 41 / 94 201 07	089 / 33 29 70
Betriebssysteme	Windows 9x, 2000, NT, XP, Linux	Windows 9x, 2000, NT, XP	Windows 98, 2000, NT, XP, Linux	Windows 9x, 2000, NT, XP, Linux	Linux	Windows 9x, 2000, NT, XP
Lieferumfang						
zusätzliche Compiler	Absoft C/C++ ¹	–	–	Fujitsu C ^{3,4}	–	Salford C/C++
Code-Kompatibilität, gemischtsprachige Programmierung	Visual C/C++ ¹ , Visual Basic ¹ , Borland C++ ¹ , Delphi ¹ , gcc/g77 ²	Visual C/C++, Visual Basic, MASM	Visual C/C++ ¹ , Intel C/C++ ¹ , g77 ²	Visual C/C++ ¹ , Visual Basic ¹ , Borland C++ ¹ , Delphi ¹ , g77 ²	k. A.	Visual C/C++ ¹ , .NET
Spracherweiterungen	VAX/VMS, IBM/VS, Sun, Cray, Lahey	VAX/VMS, Intel Fortran	VAX/VMS, Intel Fortran, Compaq Fortran	VAX/VMS, Lahey	HPF ⁶ , Fortran 2000 ⁷	–
eigene Eingabeaufforderung	✓	✓	✓	✓	✓	–
Handbücher						
gedruckt	User Guide	Getting Started, Language Reference	–	Language Reference ^{3,4} , User's Guide ^{3,4} , WISK Manual ^{3,4}	–	Getting Started
elektronisch	Fortran 90/95 Reference, Fortran 77 Reference, Support Library Guide, PLplot Reference ¹ , User Guide	Programmer's Guide, Language Reference, Developer Studio, Win32 SDK	User's Guide, Programmer's Ref. Manual, Libraries Ref. Manual	siehe oben	Readme, Technical Reports	Users Guide, Programmers Guide, ClearWin+
Man Pages ²	✓	–	✓	✓	✓	–
Entwicklungsumgebung						
IDE vorhanden / projektorient.	✓ ¹ / ✓ ¹	✓ / ✓	✓ ⁵	✓ ¹ / –	–	✓ / –
mit integriertem Debugger	✓ ¹	✓	✓ ⁵	–	–	✓
Programmibliotheken, Header-Dateien	SuperPlot ¹ , PLplot, Open GL, BLAS, LAPACK95, ATLAS, NCSA HDF, IMSL F77 ³ , IMSL F90 MP ^{3,9} , f90 sql ¹ , Sun/VAX Support ¹	PortLib, OpenGL, SciGrafik, F77 ^{3,4} , IMSL F90 MP ^{3,4}	Intel Integrated Performance NAG F77 ³ , NAG F90 ³ , IMSL Library ^{3,9} (inkl. BLAS, LAPACK, FFT, VML), IMSL F90 MP ^{2,3} , NAG F77 ^{2,3}	MS C Runtime ¹ , Fujitsu Primitives ^{3,9} , Math Kernel ^{3,4} , NAG If90 ³ , BLAS ^{2,3} , 4 ⁹ , LAPACK ^{2,9}	NAG If90 ³ , NAG F77 ³ , SSL2 ^{3,4,9} , f90 SQL Lite ¹	OpenGL
GUI-Programmierung	Win32API ¹ , MRWE ¹ , DISLIN ²	QuickWin, Win32-API	–	WISK ^{3,4} , Win32-API ¹	–	ClearWin, Win32-API
Parallelerverarbeitung						
Autoparallelsierer	✓ ⁸ (VAST)	✓ ³ (VAST)	✓	✓ ^{2,3,4}	–	–
OpenMP	✓ ⁸ (VAST)	–	✓	✓ ^{2,3,4}	–	–
Preis						
Linux / Windows	997 € / 790 € ¹⁰	– / 695 € ¹⁰	695 € / 444 € ^{5,10}	835 € / 695 € ¹⁰	995 £ / – ¹⁰	– / 657 € ¹⁰
¹ nur Windows ² nur Linux ³ gegen Aufpreis ⁴ Teil der Profi-Version ⁵ Windows-Version benötigt Visual C++ 6.0, das man getrennt kaufen muss; Preis ohne Visual C++ ⁶ nur Syntaxtest, es wird Ein-Prozessor-Code erzeugt ⁷ allokierbare Typkomponenten, IEEE-Ausnahmebehandlung ⁸ Teil von ProFortran MP ⁹ Bibliothek ist Thread-Safe ¹⁰ Rabatt für Studenten, Universitäten oder Privatpersonen verfügbar; Linux-Version des Intel-Compiler für private Nutzung kostenlos						
✓ vorhanden	– nicht vorhanden	k. A. keine Angabe				

