The Briefing Blurb:

Exploring the Ethernet with Mouse and Keyboard

1983 Edition

By Lyle Ramshaw of PARC/CSL

An immigration document in the tradition of Roy Levin's *A Field Guide to Alto-Land.*

June 7, 1983

Filed on: [Indigo]<Cedar>Documentation>BriefingBlurb.tioga, BriefingBlurb.press

Abstract: This document is a general introduction to the computing environment at PARC slanted towards the needs and interests of newcomers to the Computer Science Laboratory. If you are looking at this document on-line from within the editor named Tioga, you might want to use the level-clipping functions to see the overall structure rather than simply plowing straight through. Click the ''Levels'' button in the top menu, then click ''FirstLevelOnly'' in the new menu that appears. That will show you the major section headings. Click ''MoreLevels'' to see the subsections, or click ''AllLevels'' to read the details.

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

For Internal Use Only

Raison d'Etre

The purpose of this document is to help immigrants adapt to the local computing community. By ''the local community'', I mean primarily the Computer Science Lab, the Imaging Sciences Lab, and the Integrated Design Lab of the Xerox Palo Alto Research Center, better known by the acronyms CSL, ISL, and IDL respectively. Immigrants to other computing communities within Xerox may also find this document of interest, but I make no guarantees. I shall assume herein that said immigrants know quite a bit about computer science in general. Hence, I shall concentrate upon discussing the idiosyncratic characteristics of the local hardware environment, software environment, social environment, linguistic environment, and the like.

You will doubtless read many documents while you are at Xerox. A common convention observed in many manuals and memos is that fine points or items of complex technical content peripheral to the main discussion appear in small type, like this paragraph. You will soon discover that you cannot resist reading this fine print and that, despite its diminutive stature, it draws your eyes like a magnet. This document has such passages as well, just so that you can begin to enjoy ferreting out the diamonds hidden in the mountain of coal.

There is a great deal of useful information available on-line at Xerox in the form of documents and source programs. Reading them is often very helpful, but finding them can be a nuisance. Throughout this document, references to on-line material are indicated by *<n>*, where *n* is a citation number in the bibliography at the end of this document. Standard citations to the open literature appear as [*n*].

If you are fortunate enough to be reading this document from within Tioga (the Cedar editor), you should pause at this point to try out the ''Def'' command. If you were to select the three characters ''<n>''in the preceding paragraph and then click the ''Def'' command with the middle mouse button, you would then find yourself looking at the place in this document where ''<n>''is defined, that is, where it appears followed by a colon. You could then get back to this section of the document by clicking the ''PrevPlace'' command with any mouse button. The ''Def'' command is almost as good as an automatic indexing facility. On another topic, you might try clicking the ''FirstLevelOnly'' button (click ''Levels'' first if you can't find the ''FirstLevelOnly'' button), and then clicking ''MoreLevels'' a few times. Try scrolling a bit too. The Tioga ''levels''commands are almost as good as an automatic table of contents.

Reading a document from front to back can be mighty boring. Fortunately, this document is so disorganized that it is not at all clear that it really has a front and a back in any normal sense. You might as well just browse through and read the parts that look interesting. To help out the browsers in my reading community, I have more or less abandoned the custom of being careful to define my terms before I use them. Instead, all the relevant terms, acronyms, and the like have been collected in a separate Glossary. Some information is contained *only* in the Glossary, so you may want to skim through it later (or now, for that matter). The ''Def'' command in Tioga is particularly helpful when browsing the Glossary from within Cedar; try selecting the word ''Tioga'', and then clicking the ''Def'' button in the Glossary viewer, for example. While writing the Glossary, I assumed that you have a basic knowledge of computer science, and a modicum of common sense: don't expect to find terms like ''computer'' and ''network'' in the Glossary.

Naming Things

At the outset, you should know something about the names of the creatures that you will find here. The prevailing local philosophy about naming systems is perhaps somewhat different from the trend elsewhere. We do have our share of alphabet soup, that is, systems and languages that are named by acronyms of varying degrees of cuteness and artificiality; consider, for example: PARC, FTP, MAXC, IFS. But we are trying to avoid making this situation any worse. To this worthy end, names for hardware and software systems are frequently taken from the *Sunset Western Garden Book* [1]; Grapevine servers are named after wines; Dorados are named after capital ships; Pilot releases are named after California rivers. As this convention about names does not meet with universal approval, it seems inappropriate to offer a justification of the underlying philosophy without offering equal time to the opposition. You will doubtless provoke a far more interesting discussion if you advance your own views on naming to almost anyone wandering in the corridors.

While we are on the general topic of the names of things, we should discuss for a moment the local customs for constructing single identifiers out of multiple word phrases. Suppose that you would like to name a variable in your program ''name several words long''. In some environments, a special character that isn't a letter but that acts something like a letter is used as a word separator within identifiers; this leads to names such as

''name!several!words!long'' or ''name_several_words_long''.

No such character is in common use locally, however. Instead, shifting between upper and lower case is used to show the word boundaries, leading to the name

''NameSeveralWordsLong''.

Some people, including Don Knuth, think that identifiers with mixed case look terribly ugly. I refuse to get sucked into expressing my opinion in this document; once again, I exhort you to espouse your views in the corridors.

There are several fine points that I should mention as well. As a general rule, case is significant for identifiers in the local programming languages, but case is not significant in file names or in Grapevine R-names. Thus, the Cedar identifiers ''REF'', ''Ref'', and ''ref'' are quite distinct, but the file names ''BriefingBlurb.tioga'' and ''briefingblurb.tioga'' are equivalent, as are the R-names ''Ramshaw.PA'' and ''ramshaw.pa''. In Mesa and Cedar, there is a further convention that the case of the first letter of an identifier is used to distinguish fancy objects, such as procedures and types, from simple ones, such as integers and reals. Thus, the identifier name ''ProcWithFiveWordName'' begins with an upper case ''P'', but the name ''integerWithFiveWordName'' begins with a lower case ''i''. The latter form looks very strange to most people when they first see it. When you first tasted an olive, you probably didn't like it. Now, you probably do. Give these capitalization conventions the same chance that you would an olive.

These capitalization conventions don't work too well when acronyms and normal words appear together in one identifier. Suppose, for example, that I wanted to introduce an identifier named ''FTP version number''. Logic would demand ''FTPVersionNumber'', but this doesn't look quite right; many people would be probably write ''FTPversionNumber'' instead. Of course, since a version number is probably an integer, it should really be ''fTPVersionNumber''. Ugh. Perhaps case is being used for too many purposes?

Local Hardware

Most of the offices and some of the alcoves around PARC have personal computers in them of one flavor or another. The first of these was the Alto. There are more than a thousand Altos in existence now, spread throughout Xerox, the four universities in the University Grant program (U. of Rochester, CMU, MIT, and Stanford), and other places. In recent years, most of the local Altos have been replaced by various flavors of D-machines: Dorados, Dolphins, and Dandelions. Both D-machines and Altos come equipped with bitmap displays, mice, and Ethernet interfaces. Let's discuss these components first, and then turn our attention to the various personal computers that contain them.

Bitmap Displays

First, let's talk about displays. Different displays use different representations of images. A character display represents its image as a sequence of character codes. This is a very compact representation, but not a very flexible one; text is all you can get, and probably in only a limited selection of fonts. A vector display represents its image as a list of vector coordinates. This works very well for certain varieties of line drawings, but not so well for filled areas or text. A bitmap display, on the other hand, produces an image by taking a large matrix of zeros and ones, and putting white where the zeros are and black where the ones are (or vice versa). The great advantage of bitmap displays are their flexibility: you can specify a tremendous number of images by giving even a relatively small array of bits. Cursors and icons are two large classes of prominent examples. Of course, you do have to supply enough memory to hold all those bits. Altos and D-machines store their bitmaps in main storage. An alternative would be to provide a special chunk of memory on the side where the display's image sits; such a memory is often called a *frame buffer*.

The primary display of the Alto is a bitmap that is 608 pixels wide by 808 pixels high. Such a display is almost large enough to do a reasonable job of rendering a single 8.5" by 11" page of text. The CRT on a D-machine has the long axis horizontal instead of vertical, giving a bitmap display that is 1024 pixels wide by 808 high. It had to be 808 high so that D-machines could emulate Altos, of course. The extra space allows you to have something else on the screen as well as the somewhat scrunched page of text that you are editing.

Ere I leave you with a mistaken impression, let me note in passing that bitmap displays are not the final solution to all of the world's problems. Raster displays that can produce various levels of gray as well as black and white can depict images free of the ''jaggies'' and other artifacts that are inherent in bitmap displays [2]. And, for some purposes, color is well worth its substantial expense.

Mice

But now on to mice. A mouse has two obvious properties it rolls and it clicks. Inside the machine, the mouse position and the display cursor position are completely unrelated; but most software arranges for the cursor to ''track'' the mouse's movements. The three mouse buttons go by various names; ''left'', ''middle'', and ''right'' is one set of names. The mouse buttons are also called ''red'', ''yellow'', and ''blue'' respectively, even though physically they are nearly always black. These colorful names were proposed at an earlier time when some of the mice had their buttons running horizontally instead of vertically. Using colors (even imaginary ones!) worked better than switching back and forth between the nomenclatures ''top-middle-bottom'' and ''left-middle-right''.

Mice also come in two basic flavors: mechanical and optical. Our current mechanical mice roll on three balls: two small ones, and one large one. Motion of the large ball is sensed by two little wipers inside the mouse, one sensing side to side rolling while the other senses forward and backward rolling. The motion of each wiper drives a commutator, and little feelers slide along the commutator, producing the electrical signals that the listening computer can decode. Building one of these little gadgets is not quite as hard as building a Swiss watch, but it's in the same league. The optical mice are a more recent

innovation. An optical mouse lives on a special pad, covered with little white dots on a black background. A lens in the mouse images a portion of the pad onto the surface of a custom integrated circuit. This IC has sixteen light-sensitive regions, some of which notice that they are being shined on by the image of a white dot on the pad. As the mouse slides along the pad on its Teflon-coated underbelly, the images of the white dots move across the IC; it is subtly constructed so as to observe this phenomenon, and take appropriate electrical action. For more details on this interesting application of a custom chip, you might enjoy checking out Dick Lyon's blue-and-white report on the subject [3].

The Ethernet

Two's company, three's a network. A collection of machines within reasonable proximity is hooked together by an Ethernet; if that doesn't sound familiar, I know of some blue-and-whites that you might like to browse [4,5]. Ethernets are connected to each other by Gateways and phone lines, which for most purposes allow us to ignore the topology of the resulting network. The resulting network as a whole is called an *Internet*. Occasionally, it's nice to know where things *really* are, and that's when a map <6> is helpful.

Ethernets come in two flavors: old and new. The old one runs at 3 MBits/sec, and should now be referred to as the ''Experimental Ethernet''. The unqualified name ''Ethernet'' should be reserved for the new one, the standardized version used in OSD products; it runs at 10 MBits/sec.

We all know how uncommunicative computers can be when left to their own devices. That's why we invent careful protocols for them to use in talking to each other. There are two entire worlds of protocols that are spoken on our various Ethernets as well: old and new. The old ones are called PUP-based (PARC Universal Packet) [7]. The new ones are known by the acronym NS (Network Systems) [8, 9]. I'm sure that the NS protocols must be documented, but I don't know where; sorry. Each protocol world includes a hierarchy of protocols for various purposes such as transporting files, or sending and receiving mail.

In addition to connecting up all of the personal computers, the network also includes a number of machines generically called *servers*. Normally, servers have special purpose, expensive hardware attached to them, such as large-capacity disks, or printers. Their purpose in life is to make that hardware available to the local community. We tend to identify servers by function, so we talk about print servers, file servers, name lookup servers, mailbox servers, tape servers, and so on. Many of the protocols for use of the Ethernet were developed precisely so that personal computers could communicate effectively with servers.

The Alto

The innards of the Alto are wonderfully described in a clear and informative blue-and-white report [10]; I seriously recommend that you read it. In the very unlikely event that you need to know still more about the Alto, you might try looking in the Alto hardware manual <11>. But for our purposes, suffice it to say that the Alto is a 16-bit minicomputer whose primary claim to fame is that it comes equipped with a bitmap display, a mouse, and an Ethernet interface.

D-Machines

The D-machines are a family of personal computers, each member of which has a name starting with the letter ''D''. As long as you don't look too closely, D-machines look a lot alike. In particular, they are all 16-bit computers with a microprogrammed processor that handles most of the I/O as well running the user's programs. And they all generally come equipped with a hard disk, a bitmap display, a keyboard, a mouse, and an Ethernet interface. There are differences of course: in size, in speed, and in flexibility.

The Dolphin (formerly called the D0)

The Dolphin was one of the early D-machines, and there are still some of them around. Dolphins are housed in the same sized chassis as Altos. You can tell that they aren't Altos because they have wide screen terminals, and because they don't have a slot on top for a removable disk pack. Instead, they use a 28MByte Winchester disk drive made by Shugart. Dolphins can talk to both 3 MBit and 10 MBit Ethernets.

The Dandelion

The Dandelion is the D-machine processor that is used in the Star products. It comes in a box about half the width of an Alto chassis, and roughly the same height and depth. Dandelions are less flexible than Dolphins, since the microprocessor is shared among the various I/O devices and the emulator in a fairly rigid round-robin fashion (associated with the terms ''clicks'' and ''rounds''). As a consequence, it isn't very easy to hang a new I/O device off of a Dandelion. On the other hand, Dandelions are both faster and cheaper than Dolphins. Dandelions talk only to 10 MBit Ethernets.

The Dorado

Building large software systems is a demanding chore. It doesn't help any when the hardware upon which your programming environment is based doesn't have enough horsepower to support you properly that is, in the manner to which you would like to become accustomed. After some years of trying to shoehorn large programs into Altos, CSL twisted the arms of its hardware folk and talked them into building the Dorado, the current high-performance model in the D-machine line. The processor, the instruction fetch unit, and the memory system of the Dorado have been written up in papers for your enjoyment [12]. Dorados come equipped with an 80 MByte removable-pack disk drive at present; new models may start showing up soon with a 315 MByte Winchester drive instead. Dorados talk only to 3 MBit Ethernets at present.

A Dorado is roughly three to five times faster than an Alto when emulating an Alto, that is, running BCPL. A Dorado runs compute-bound Mesa software roughly eight to ten times as fast as an Alto. Because of the raw power of a Dorado, it is usually the computer of choice for substantial programming projects. The primary difficulty about Dorados is that there aren't enough of them (and the related fact that they are rather tricky to build). Some people have their own, but others must share a pool of public machines. Now, even though the Dorado disk drives have removable packs, it really isn't very convenient to start your session of a public Dorado by mounting your own pack. The biggest difficulty is that you must be at the processor to change the disk pack, and the processor is a long way away. Subsidiary difficulties are that you must power down a Dorado in order to change the disk pack, and that T-80 disk packs are difficult to label effectively. As a result, when you borrow a Dorado, you generally also want to borrow at least some of the space on that Dorado's local disk. In order for this sharing to work out well, certain social taboos and customs concerning the use of such local disks have emerged, under the general rubric of ''living cleanly''. More on this topic anon.

In a return to the ways of the past, the Dorado processors are rack mounted in a remote, heavily air-conditioned machine room. It was initially intended that the Dorado, like the Alto, would live in your office. To prevent its noise output from driving you crazy, a very massive case was designed, complete with many pounds of sound-deadening material. But experience indicated that Dorados ran too hot when inside of these cabinets, and the concept of having Dorado processors in offices was abandoned. With progress in general and VLSI in particular, there is hope that some successor to the Dorado will once again come out of the machine room and into your office.

The Dicentra

The Dicentra is the newest D-machine. Essentially, it consists of the processor of the Dandelion with the tasking stuff striped out squeezed onto one Multibus card. It communicates with its memory and with I/O devices over the Multibus. Dicentras will talk to any Ethernet, or any I/O device for that matter, for which you can supply a Multibus interface card; that's one of the Dicentra's strengths. The initial application of the Dicentra is as a processor for low cost Internet gateways. The Dicentra and the Dandelion are named after wildflowers partially because they are outgrowths of an initial design of Butler Lampson's called the Wildflower.

The Daffodil

The Daffodil is a D-machine that doesn't exist yet, but someday may. If so, it will be cheap to build, since it will use custom integrated circuits. The Daffodil is product-related. Thus, please don't talk about it too widely. I mention it hear only so that you will know what it is that Chuck Thacker is talking about.

The Dragon

The Dragon is a high-performance processor based on custom integrated circuits that is being designed in CSL; confusingly enough, though, the Dragon is not really a D-machine. For example, the Dragon word size is 32 bits rather than 16. The underpinnings of Cedar will be adjusted as necessary so that Cedar will run on a Dragon; but this will take some doing.

A few comments about Booting

All of the local processors come equipped with a hidden button called the ''boot button'' that is used to reinitialize the processor's state. The Alto had just one boot button, hidden behind the keyboard; pushing it booted the Alto. On Dolphins, the situation is only slightly more complex: there are two boot buttons, one at the back of the keyboard, and the other on the processor chassis itself. They perform roughly the same function, but the one on the chassis is a little more potent. On Dorados, there is a lot more going on. There are really two computers involved, the main Dorado processor and a separate microcomputer called the *baseboard.* It is the baseboard computer's job to monitor the power supplies and temperature and to stage-manage the complex process of powering up and down the main processor, including the correct initialization of all of its RAM's. The boot button on a Dorado is actually a way of communicating with this baseboard computer. You encode your request to the baseboard computer by pushing the boot button repeatedly: each number of pushes means something different. For details, see Ed Taft's memo on the subject <13>. If the baseboard computer of the Dorado has gone west for some reason (as occasionally happens), your only hope is to push the *real* boot button, a little white button located on the processor chassis itself, far, far away. Just as the boot button on the keyboard is essentially a one-bit input device for the baseboard computer, the baseboard computer also has a one-bit output device: a green light located on the processor chassis. Various patterns of flashing of this light mean various things, as detailed in <13>.

There is one more bit of folklore about booting that I can't resist mentioning every once in a while, I have to throw in some subtle tidbit to keep the wizards who read this from getting bored. Our subject this time is the ''long push boot''. Suppose that you have been working on your Dorado for a while, and you walk away to go to the bathroom. When you return and reach toward your keyboard, you get a static shock. You are only mildly annoyed at this until you notice that the cursor is no longer tracking the mouse, and the machine doesn't seem to hear any of your keystrokes. The screen looks OK, but the Dorado is ignoring all input. What has probably happened is that the microprocessor in your terminal has been knocked out by the static shock. Yes, Virginia! In addition to the Dorado itself, and the baseboard computer, there is also a microprocessor in your terminal (located in the display housing), which observes your input actions and sends them on to the main processor under a protocol referred to as ''the seven-wire interface''. What you want to do now is to reboot the terminal microprocessor without disturbing the state of the Dorado at all after all, you were in the process of editing something,

and you are now in danger of loosing those edits. What you should do is to depress the boot button and hold it down for quite a while (more than 2.5 seconds); and then release it. This is known as a ‘‘long push boot’’, and it does just what you want under these conditions: it reboots your terminal without affecting anything higher up.

MAXC: a blast from the past

Before we leave the topic of hardware completely, I should pause to mention the existence of MAXC (the name is said to be an acronym for Multiple Access Xerox Computer). Over the years, the folk in CSL built two MAXC’s. Each was a good-sized microprogrammed computer that spent its days emulating a PDP-10: running TENEX, and timesharing away with the best of them. One of the MAXC’s still survives, the one initially known as MAXC2, and it serves us now primarily as the Internet’s interface to the Arpanet. Vestiges of MAXC1 still survive as souvenirs in some people’s offices. Most of the stuff going between the Internet and the Arpanet is electronic mail; our mail systems understand about Arpanet recipients, so there is no need to talk to MAXC directly just to send Arpanet mail. There are a few other computing tasks that MAXC can perform and that no one has yet had the energy to supply in some other way, such as archiving files onto magnetic tape. But most folks should be able to spend their time here quite happily without ever talking directly to MAXC.

Local Programming Environments

Various programming environments have grown up around the various pieces of hardware mentioned above. You can get a software merit badge simply by writing one non-trivial program in each envirnoment.

Programming on MAXC

Since we were discussing MAXC just a moment ago, let's get it out of the way first. From a software point of view, MAXC is a PDP-10. Thus, it is programmed either in the assembler Macro-10 or else in one of a variety of higher level languages [14]. Fortunately, there aren't all that many new programs that have to be written to run on MAXC any more.

BCPL

The first high-level programming language used on the Alto was BCPL, and quite a bit of program writing was done in that environment over the years. By now, however, essentially no new programming is being done in BCPL. The language itself will be around for some time to come, since there are BCPL programs that perform valuable services for us: the print server programs Press and Spruce and the file server program IFS are three important examples.

Of the better-known computer languages, BCPL is closest to C. The fundamental data type in BCPL is a sixteen-bit word. There are facilities in the language for building structured data objects including records and pointers. But there is no type-checking in the language at all. For example, if *foo* is a pointer to a record of type *node* that includes a field named *next*, that field is referenced in BCPL by writing

''foo>>node.next'',

which means ''treat *foo* as a pointer to a *node*, and extract the *next* field''. In a strongly typed language, you wouldn't have to mention that *foo* was a pointer to a *node*, since the compiler would be keeping track of the fact that *foo* was so declared. The BCPL compiler, however, thinks of *foo* as a sixteen bit value, just like any other sixteen bit value. For example, it would be legal in BCPL to write

''(foo+7)>>node.next'', or ''foo>>otherNode.next''.

Some of the strictness of the Mesa approach to type-checking and version matching discussed below may be a reaction to BCPL's free-wheeling ways of handling these issues. Further details about the BCPL language and environment can be found elsewhere <14, 15, 16>.

The debugger in the BCPL environment was named ''Swat''. This name is preserved in the local dialect as the name of the bottom of the three unmarked keys at the right edge of the keyboard. Various debuggers may be invoked in various environments by depressing this key, perhaps in conjunction with the left-hand shift key or the control key. (The right hand shift key won't do; it is too close to the swat key itself for comfort!)

Mesa

Mesa is a strongly typed, PASCAL-like implementation language designed and built locally. It first ran on Altos. Herein, I shall call that system Alto/Mesa. Dolphins and Dorados (but not Dandelions) can run Alto/Mesa by impersonating an Alto at some level. More recent instances of Mesa now run on all of our D-machines under the Pilot operating system. In passing, I should observe that Pilot is an operating system written in Mesa by folk in SDD. It is a heavier-weight operating system than the Alto OS, providing its clients with multiprocessing, virtual memory, and mapped files.

Alto/Mesa programs do not use the Alto OS at all, mostly because Mesa and BCPL have rather

different philosophies about the run-time world in which they exist. So the first thing that a Mesa program does when running on an Alto is to junta away almost all of the OS, and set about building a separate Mesa world. It is a considerable nuisance for Mesa and BCPL programs to communicate, since their underlying instruction sets are completely different. So, most of the important OS facilities, such as the file system, had to be re-implemented directly in Mesa. Mesa's memory management strategies replace the revolutionary tactics of ''junta'' and ''counter-junta'' with the relative anarchy of segment swapping.

A fair amount of software was written in Alto/Mesa, but little new programming is being done in that environment; that is, Alto/Mesa isn't quite at dead as BCPL, but it is getting there. Perhaps the crown jewels of Alto/Mesa are the systems Laurel, Grapevine, Mockingbird, and Griffin. You will be hearing more about the former two in the section on electronic mail; to find out more about the latter two, check out their entries in the Glossary.

The Pilot version of Mesa is the home to lots of active programming in several locations. First, it is the system in which the Star product was and is being implemented by OSD. The programmers in OSD have developed a set of tools for programming in Mesa variously called the ''Tools Environment'' or ''Tajo''. This body of software may soon be marketed under the name ''the Mesa Development Envirnoment''. In addition, Pilot Mesa is the current base of the Cedar project in CSL and ISL. More on Cedar later.

Although Mesa programs look a lot like PASCAL programs when viewed in the small, Mesa provides and enforces a modularization concept that allows large programs to be built up out of smaller pieces. These smaller pieces are compiled separately, and yet the strong type checking of Mesa is enforced even between different modules. The basic idea is to structure a system by determining certain abstract collections of facilities that some portions of the system will supply to other portions. Such an abstraction is called an ''interface'', and it is codified for the compiler's benefit in a Mesa source file called an ''interface module''. An interface module defines certain types, and specifies a collection of procedures that act on values of those types. Only the procedure headers go into the interface module, not the procedure bodies (except for INLINE's, sad to say). This makes sense, since all the interface module has to do is to give the compiler enough information so that it can type-check programs that use the abstraction.

Having specified the interface, some lucky hacker then has the job of implementing it that is, of writing the procedure bodies that actually do the work. These procedure bodies go into a different type of module called an ''implementation module''. An implementation module is said to ''export'' the interface that it is implementing; it may also ''import'' other interfaces that it needs to do its job, interfaces that some other program will implement.

In simple systems, each interface is exported by exactly one module. In such a system, there isn't much question about who should be supplying which services to whom. In fact, in these simple cases, the *binding*, that is, the resolution of imports and exports, can be done on the fly by the loader. But in more complex cases, there might be several different modules in the system that can supply the same service under somewhat different conditions, or with somewhat different performance. Then, the job of describing exactly which modules are to supply which services to which other modules can become rather subtle. A whole language was devised to describe these subtle cases, called C/Mesa. The Binder is the program that reads a C/Mesa description, called a *config*, and builds a runnable system by filling imports request from exports according to the recipe.

The Mesa language is described by a manual [18]. It lies somewhere between a tutorial and a reference manual. Some people find some portions of it rather obscure; in particular, the discussion of interfaces and implementations in Chapter 7 is often cited as confusing. To make matters a little worse, that manual documents Mesa version 5.0; the current Alto/Mesa is version 6.0, and Pilot mesa has advanced even further. From the point of view of the Mesa language itself, the most important changes that have occurred since version 5.0 are the introduction of sequences and zones in version 6.0; they are documented for your reading pleasure <19>. You may also be interested in Jim Morris's comments

on how programs should be structured in Mesa [20].

Smalltalk

Smalltalk was developed by the folk who now call themselves the Software Concepts Group (formerly known as the Learning Research Group). The Smalltalk language is the purest local embodiment of ''object-oriented'' programming:

A computing world is composed of ''objects''.

The only way to manipulate an object is to be polite, and ask it to manipulate itself. One asks by sending the object a message. All computing gets done by objects sending messages to other objects.

Every object is an ''instance'' of some ''class''.

The class definition specifies the behavior of all of its instances that is, it specifies their behavior in response to the recipt of various messages.

Genealogists will recognize that ideas from both Simula and Lisp made their way into Smalltalk, together with traces of many other languages.

For some years now, the folk in SCG have been working at trying to get the Smalltalk language and system out into the great wide world. The first public event that came out of this effort was the August 1981 issue of Byte magazine; it was devoted to Smalltalk-80, including a colorful cover drawing of the now famous Smalltalk balloon. In addition, the SCG folk are writing several books about Smalltalk, and they are planning to license the system itself to various outside vendors. The first of the books, entitled *Smalltalk-80: The Language and Its Implementation*, emerged from the presses at Addison-Wesley just recently [21]. Future books will include *Smalltalk-80: The Interactive Programming Environment*, and *Smalltalk-80: Bits of History, Words of Advice*.

Interlisp-D

LISP is the standard language of the Artificial Intelligence community. Pure LISP is basically a computational incarnation of the lambda calculus; but the LISP dialects in common use are richer and bigger languages than pure LISP. Interlisp is one dialect of LISP, an outgrowth of an earlier language called BBN-LISP; for more historical details, read the first few pages of the Interlisp Reference Manual [22]. One of the biggest strengths of Interlisp is the large body of software that has developed to assist people programming in Interlisp. Consider the many features of Interlisp: an interpreter, a compatible compiler, sophisticated debugging facilities, a structure-based editor, a DWIM (Do What I Mean) error correction facility, a programmer's assistant, the CLISP package for Algol-like syntax, the Masterscope static program analysis database, and the Transor LISP-to-LISP translator, to name a few.

Interlisp itself has been implemented several times. Interlisp-10 is the widely-used version that runs on PDP-10's. Interlisp-D is an implementation of Interlisp on the D-machines [23], produced by folk at PARC. In the process of building Interlisp-D, the boundary between Interlisp and the underlying virtual machine was moved downward somewhat, to minimize the dependencies of Interlisp on its software environment; that is, functions that were considered primitive in Interlisp-10 were implemented in Lisp itself in Interlisp-D. But the principal innovations of Interlisp-D are the extensions that give the Interlisp user access to the personal machine computing environment: network facilities and high-level graphics facilities (including a window package) among them.

By the way, Interlisp has the honor of being the first system (to my knowledge) to use the prefix ''Inter-''. This prefix has become quite the rage of late: Internet, Interpress, Interscript you get the general idea.

Cedar

Back in 1978, folk in CSL began to consider the question of what programming environment we would use on the emerging D-machines. A working group was formed to consider the programming environments that then existed (Lisp, Mesa, and Smalltalk) and to form a catalog of programming environment capabilities, ranked by both by value and by cost. A somewhat cleaned-up version of the report of that working group is available as a blue-and-white for your perusal [24]. After pondering the alternatives for a while, CSL chose to build a new programming environment, based on the Mesa language, that would be the basis for most of our programming during the next few years. That new environment is named ''Cedar''.

Cedar documentation is in a constant state of flux; indeed, it might be said that Cedar as a whole, not only its documentation, is in a constant state of flux. Much of the documentation for the current release is accessible through a ''.df'' file named Manual.df <25>. Hardcopies of this packet of stuff, entitled ''The Cedar Manual'', are produced from time to time, and distributed to Cedar programmers.

The programming language underlying Cedar is essentially Mesa with garbage collection added. Now, adding garbage collection actually changes things quite a bit. First of all, it changes programming style in large systems tremendously. Without garbage collection, you have to enforce some set of conventions about who owns the storage. When I call you and pass you a string argument, we must agree whether I am just letting you look at my string, or I am actually turning over ownership of the string to you. If we don't see eye to eye on this point, either we will end up both owning the string (and you will aggravate me by changing *my* string!) or else neither of us will own it (and its storage will never be reclaimed a storage leak). Once garbage collection is available, most of these problems go away: God, in the person of the garbage collector, owns all of the storage; it gets reclaimed when it is no longer needed, and not before. But there is a price to be paid for this convenience. The garbage collector takes time to do its work. In addition, all programmers must follow certain rules about using pointers so as not to confuse the garbage collector about what is garbage and what is not.

Thus, programs in the programming language underlying Cedar look a lot like Mesa programs, but they aren't really Mesa programs at all, on a deeper level. To avoid confusion, we decided to use the name ''Cedar'' to describe the Cedar programming language, as well as the environment built on top of it. Cedar is really two programming langauges: a restricted subset called the *safe language*, and the unrestricted full language. Programmers who stick to the safe language can rest secure in the confidence that nothing that they can write could possibly confuse the garbage collector. Their bugs will not risk bringing down the entire environment around them in a rubble of bits. Those who choose to veer outside of the safe language had better know what they are doing.

Those who want to know more about Cedar are once again encouraged to dredge up a copy of the Cedar Manual <25>. It includes documentation on how Cedar differs from Mesa, annotated examples of Cedar programs, manuals for many of Cedar's component parts, a Cedar catalog, and lots of other good stuff. By the way, the most authoritative source for what the current Cedar compiler will do on funny inputs can be found in a document called the Cedar Language Reference Manual, also known by the acronym CLRM. This is logically part of the Cedar Manual, but it is currently bound separately, and only available in draft form. The CLRM suggests a particular design philosophy for building a polymorphic language that is a superset of the current Cedar, since that is the direction in which the authors of the CLRM, Butler Lampson and Ed Satterthwaite, would like to nudge the Cedar language.

Local Software

This section is a once-over-lightly introduction to some of the major software systems that are available in the Alto and Cedar worlds. First, let me mumble some general words about how such subsystems are documented. The most commonly used Alto subsystems are documented in a tome called the Alto User's Handbook [26]. The less commonly used ones are documented in a catalog entitled ''Alto Subsystems'' <27>. In addition, Suzan Jerome wrote a Bravo primer aimed at non-programmers [28]. In Cedar, the current best sources are the Cedar Manual mentioned above <25>, and a brand new public database, sitting on Alpine, containing whiteboards of Cedar documentation. Unfortunately, I won't hear about the latter until Dealer tomorrow, so that I can't tell you any more about it at the moment; I'm sorry, but that's life in a rapidly changing world. Wow! I've seen the whiteboards stuff now, and it's flashy! Maybe this is the last version of the Briefing Blurb that I'll ever have to write.

Filing

When programming in the Alto world, or in current Cedar, you are dealing with two different types of file systems: local and remote. The local file system sits on your machine's hard disk. Remote file systems are located on file servers, machines with big disks that are willing to store files for you. Local file systems have several unpleasant characteristics in comparison with the remote systems: they are small, and they aren't very reliable. Both of these problems have consequences.

Because local file systems are small, it isn't in general practical to store more than one version of a file on the local disk. Thus, in our current local file systems, writing a ''new version'' of a file really means writing on top of the old one. Nearly everyone who isn't accustomed to this (particularly PDP-10 hackers) gets burned by it at least once. There is one important exception to this general rule of ''no old versions'', however: our text editors maintain one backup copy of each file being edited as a separate file, whose name ends with a dollar sign. That is, the backup copy of ''foo.tioga'' is stored in the file ''foo.tioga$'', and similarly for Bravo. Note that our remote file servers do maintain multiple versions of files. Letting old versions of things accumulate is one easy way to overflow your disk usage allocation on a remote server.

No disk is completely reliable. Our remote file servers have automatic backup facilities that protect us from catastrophic disk failures. But the local file systems have no such automatic protection. Since this protection isn't provided automatically, it behooves you to adjust your behavior appropriately: make sure that, on a regular basis, backup copies of the information on your local disk are put in some safe place, such as on a remote file server where suitable precautions are constantly being taken by wizards to protect against disk failure. Doing this is one facet of what is meant by the phrase *Living Cleanly*, which deserves its own section.

Living Cleanly (also known as ''Keeping your bags packed'')

The phrases ''living cleanly'' and ''keeping your bags packed'' refer to a particular style of use of your local file system. In order to understand the cosmic issues involved, we should pause to discuss the ways in which local and remote file systems have been used over the years.

Back in the Alto days, personal files were usually stored on one's Alto disk pack, while project-related and other public files were stored on remote servers. Careful folk would occasionally store backup copies of their personal files on remote servers as well, in case of a head crash. But, as a general rule, one thought of one's Alto pack as the repository of one's electronic state. This made sharing Altos quite convenient, since you could turn any physical Alto into ''your Alto'' just by spinning up your disk pack.

In the glorious world of the Cedar future, all of your personal files as well as all public files will live on file servers in the network. The disk attached to your personal computer will, from time to time, contain copies of some of this network information, for performance reasons; but you won't have to do

anything to achieve this, and you won't have to worry about how it is done. From the user's point of view, all files will act as if they were remote at all times. Indeed, except in a few funny cases, there won't even be any notion of ''local file''; ''file'' will mean ''remote file''.

At the moment, we are sitting in an unpleasant trasitional phase somewhere between these two styles of usage of the local disk: we are attempting to simulate the latter state by means of manual methods and social pressure. We want you to think of your data as really living out on the file servers. That is the proper permanent home for your personal files as well as for public files. You will have to bring copies of these files, both private and public, to your local disk in order to work on them. But, at the end of each editing session, you should store the new versions of files that you have created back out to their permanent remote homes. None of this happens automatically at present; you have to make it happen manually by using various file shuffling tools, such as the ''DF files'' discussed below. Using these tools is a hassle, and learning how to use them can be confusing. But, there are four important benefits to be reaped from adopting a clean living life-style.

First, you are taking a step towards the glorious future.

Secondly, you are protecting yourself against failures of the local disk. A clean liver only holds information on her local disk for the duration of an editing session. This puts a reasonable bound on the amount of information that she can lose because of a disk crash.

Thirdly, there are various reasons why erasing your local disk is a good idea when updating to a new release of the Cedar system; sometimes, in fact, it is required. Since clean living folk don't keep long term state on their local disks, this doesn't bother them in the slightest.

Finally, and perhaps most importantly, clean living is the key to sharing disk space on machines without removable disks. When you use a public Dolphin or Dorado, you are forced to share its disk space with the other members of the community. This sharing is predicated on a policy of clean living: when your session is over, you must store away all of your files on remote file servers. The person who uses the machine next may need to free up some disk space; if so, she is perfectly entitled to delete your files without qualm or pause. And you won't mind a bit, it says here, because you have been living cleanly.

The above paragraph is the ''letter of the law'' regarding the sharing of public disk space. People who want to be well regarded should also pay some attention to the ''spirit of the law'': sharing things is always more pleasant when everyone acts with a modicum of politeness and care. Don't delete the previous user's files if she was called away by some disaster and didn't have a chance to clean up. Try not to delete the standard system files, such as the Compiler, that sit in the local file system, since whoever follows you will be justifiably aggravated by their absence. Even more important, if you do exotic things such as bringing over non-standard versions of system files, try to put everything back to normal when you leave ere you cause whoever follows you to become hopelessly confused.

Local file systems

The local file system in the Alto world is called either the ''Alto file system'' or the ''BFS'', the latter being an acronym for Basic File System. The biggest that a BFS can be is 22,736 pages. This is substantially bigger than the entire disk on an Alto. However, Dolphins and Dorados have much bigger local disks. Hence, when a Dolphin or Dorado is emulating an Alto, its local disk is split up into separate worlds called *partitions*, each containing a maximum-sized BFS. Dolphin disks can hold two full partitions, while Dorado disks can hold five. What partition you are currently accessing is determined by the contents of some registers that the disk microcode uses. There is a command called ''partition'' in the Executive and the NetExec that allows you to change the current partition.

When operating in the Pilot world, a disk pack is called a physical volume, and it is divided into worlds called logical volumes. (Pilot, you will recall, is the new operating system written in SDD.) The

area of the disk devoted to Pilot volumes must be disjoint from the area devoted to Alto-style partitions. Most Dolphins that run Cedar are set up with a half-sized Alto partition, and the other three-quarters of the disk devoted to Pilot; most Dorados that run Cedar have one full-sized Alto partition, and the other four-fifths of the disk devoted to Pilot.

In current Cedar, many programs still restrict you to working with files in the local file system, which is maintained by Pilot in the appropriate logical volume. The editor Tioga, for example, will let you read remote files specified by a full path name, but it won't let you edit them; only local files may be modified. In subsequent Cedar's, there will be a new local file system and directory package, the Nucleus and FS respectively, to go along with the new virtual memory manager (also part of the Nucleus). These wonders will make it somewhat easier to ignore the existence of the local file system, except for its beneficial effects on performance; that is, they will make clean living more nearly automatic.

All of our local file systems use a representation for files that drastically reduces the possibility of a hardware or software error destroying the disk's contents. The basic idea is that you must tell the disk not only the address of the sector you want to read or write, but also what you think that sector holds. This is implemented by dividing every sector into 3 parts: a header, a label, and a data field. Each field may be independently read, written, or compared with memory during a single pass over the sector. The Alto file system stuffs a unique identification of the disk block, consisting of a file serial number and the page number within the file, into the label field. Now, when the software goes to write a sector, it typically asks the hardware to compare the label contents against data in memory, and to abort the writing of the data field if the compare fails. This makes it pretty difficult, though not impossible, to write in the wrong place. Furthermore, it distributes the structural information needed to reconstruct the file system over the whole disk, instead of localizing it in one place, the directory data structures, where a local disaster might wipe it out. Each local file system also has a utility program called a Scavenger that rebuilds the directory information by looking at all of the disk labels.

Remote file systems

The most important local file servers are IFS's, an acronym for Interim File System (one of the crown jewels of the BCPL programming environment). Like I always say, ''temporary'' means ''until it breaks'', and ''permanent'' means ''until we change our minds''. Indigo and Ivy are two prominent local IFS's; Indigo stores mostly project files, while Ivy stores mostly personal files. MAXC also serves as a file server for some specialized applications. Juniper was CSL's first attempt to build a distributed transactional file server; it was one of the first large programs written in Mesa. Alpine is a new effort to build such a beast in the context of Cedar, in support of distributed databases and other such wonderful things. Some Walnut users have been storing their mail databases on Alpine for a month or more.

There is no coherent logic to the placement of ''general interest'' files and directories, nor even to the division between Maxc, Indigo, and Ivy. Browse through the glossary at the end of this document to get a rough idea of what's around. If something was made available to the universities in the University Grant program, then it is probably on Maxc (or archived off of Maxc), since Maxc is the machine that the university folk can access.

IFS supplies a general sub-directory structure which the Maxc file system lacks, and as a result there are lots of place to look for a file on an IFS. For example, on Maxc you might look for

[Maxc]<AltoDocs>MyFavoritePackage.press

while on IFS you would probably look for

[Indigo]<Packages>Doc>MyFavoritePackage.press, or

[Indigo]<Packages>MyFavoritePackage>Documentation.press,

or perhaps some other permutation. This requires a bit of creativity and a little practice. However, if you get in the habit of using ''*''sin file name specifications, you will find all sorts of things you might

not otherwise locate. Note that a ‘‘*’’in a request to an IFS will expand into all possible sequences of characters, *including* right angle brackets and periods. Thus, for example, a request for

<Packages>*press

refers to all files on all subdirectories of the Packages directory that end with the characters ‘‘press’’. A ‘‘*’’won’t match a left angle bracket, by the way. Thus, if you ask for ‘‘*.press’’, you are referring to all Press files on the current directory. If you ask for ‘‘<*.press", you are referring to all of the Press files on the entire IFS (expect such a search to take a long time!).

Warning: Once you have gotten used to the IFS conventions about ‘‘*’’s in file names, you will find the TENEX rules quite restrictive and unnatural. On TENEX, asterisks can be used for only two purposes: either to wildcard the entire prefix of the filename or to wildcard the entire extension. If you want to refer to all of the files on a TENEX directory, you must say ‘‘*.*’’, not just ‘‘*’’; if you want to refer to all of the files whose names start with an ‘‘H’’, you are simply out of luck. This lack of ‘‘forward compatibility’’ (the opposite of backward compatibility?) has tripped up many a searcher.

There is a movement afoot in the Cedar world to simplify our file naming conventions by replacing the various flavors of brackets with a UNIX-like slash. Thus, in some Cedar systems, such as the FileTool, the documentation file mentioned above could be referred to as

/Indigo/Packages/MyFavoritePackage/Documentation.press.

File Properties

The ‘‘size’’of a file is its length measured in disk pages; the ‘‘length’’ of a file is its length measured in bytes. The ‘‘create date’’ of a file is the date and time at which the information in that particular version of the file was ‘‘created’’, that is, the date when this that sequence of bytes came into being. Copying a file from one file system to another does not change the create date, since the information in the file, the sequence of bytes, is not affected. The create date is almost always what you want to know about a file. Some of our systems also maintain a ‘‘write date’’ or a ‘‘read date’’, but they are less well defined, and not as interesting.

Editing and Typesetting

In the outside world, document production systems are usually de-coupled from text editors. One normally takes the text that one wants to include in a document, wraps it in mysterious commands understood by a document processor, feeds it to that processor, and puzzles over the resulting jumble of characters on the page. In short, one programs in the document processor’s language using conventional programming tools an editor, a compiler, and sometimes even a debugger. Programmers tend to think this is neat; after all, one can do anything with a sufficiently powerful programming language. (Remember, Turing machines supply a sufficiently powerful programming language too.) However, document processors of this sort frequently define bizarre and semantically complex languages, and one soon discovers that all of the time goes into the edit/compile/debug cycle, not careful prose composition.

Bravo is the editor and typesetter in the Alto world, and it represented a modest step away from the programming paradigm for document production. A single program provided both the usual editing functions *and* a reasonable collection of formatting tools. You can’t program Bravo as you would a document ‘‘compiler’’, but you can get very tolerable results in far less time. The secret is in the philosophy: what you see on the screen is what you get on paper. You use the editing and formatting commands to produce on the screen the page layout you want. Then, you tell Bravo to ship it to a print server and presto! You have a hardcopy version of what you saw on the screen. Sounds simple, right?

Of course, it isn’t quite that easy in practice. There are dozens of subtle points having to do with fonts, margins, tabs, headings, and on and on. Bravo was a success because most of these issues are

resolved more or less by fiat someone prepared a collection of configuration parameters and a set of forms that accommodated most document production. Many of the configuration options aren't even documented, so it is hard to get enough rope to hang yourself. The net effect is that one spent more time composing and less time compiling.

In Bravo's wake, several new editors of unformatted text appeared: the Laurel editor, and the editor in the Tools Environment are prominent examples. The Laurel editor is particularly noteworthy in that it pioneered the development of a modeless (or at least less modal) user interface for an editor. The Star product editor and Tioga are more recent local editors in the full Bravo tradition: they can handle formatting and multiple fonts. Tioga is the editor within Cedar, and its user interface is very close to the widely beloved Laurel modeless interface try going back to Bravo after using Tioga for a while, and see how horrible it feels to have to remember to type ''i'' and ''ESC'' all the time. Tioga shows formatted text on the screen. To get a hardcopy of that text, the current path involves running a companion program called the TSetter, which will compose your pages for printing and send them to a print server. Tioga's documentation is particularly convenient, since it usually available in iconic form at the bottom of the Cedar screen <29>.

Dealing with editor bugs

All text editors have bugs. Furthermore, you are often most likely to tickle one of the remaining bugs in an editor when you are working furiously on a hard problem, and hence, have been editing for a long time without saving the intermediate results. As fate would have it, these are exactly the times when it is most damaging and most upsetting to lose your work. There is nothing quite like the sinking feeling you get when a large number of your precious keystrokes gurgle away down the drain. Both Bravo and Tioga have mechanisms that can, in some cases, save you from the horrible fate of having to do all those hours of editing over again. Bravo attempts to safeguard you by keeping track of everything that you have done during the editing session in a log file; in case of disaster, this log can be replayed to recapture most of the effects of the session. If you have a disaster when editing in Bravo, be careful NOT to respond by running Bravo again to assess the damage. By running Bravo again in the normal way, you will instantly sacrifice all chance of benefiting from the log mechanism, since the log allows replay only of the most recent session. What you want to do instead is run the program ''BravoBug'' (''Bravo/R'' is not an adequate substitute). It wouldn't be a bad idea to ask a wizard for help also. While you are looking for a wizard, try and think of some good answer to the question ''Why are you using Bravo, anyway?'', which said wizard will almost certainly ask.

The most common perhaps I should really say ''the least rare'' source of editing disasters in Tioga is problems with monitor locks. Unfortunately, this class of problem usually makes further progress in any part of Cedar impossible, since Tioga is so basic to the Cedar system. If you can get to the CoCedar debugger, you might be able to save your edits by calling the procedure

_ ViewerOpsImpl.SaveAllEdits[ ]

Rumor has it that Cedar versions from 4.2 on will allow you to invoke this procedure by hitting a special collection of keys in Cedar itself, even after Tioga has become wedged. A further rumor has supplied more details: holding down both the left and the right shift keys and the Swat key for more than 1 second will invoke SaveAllEdits[ ]. While the saving is taking place, the cursor will become a black box.

Printing

In general, our printers are built by taking a Xerox copier and adding electronics and a scanning laser that produce a light image to be copied. There are many different types of such printers, and there are multiple instances of each printer type as well. There are also many different programs that would like to produce printed output. The Press print file format was our first answer to the problem of allowing every printing client to use every printer. Press files are the Esperanto of printing. Most print

servers demand that the documents that you send to them be in Press format. This means you have to convert whatever you have in hand (often text) to Press format before a server will deign to print it.

Press file format <30> is hairy, and some print servers don't support the full generality of Press. Generally, however, such servers will simply ignore what they can't figure out, so you can safely send them any Press file you have.

A Press file can ask that text be printed in one of an extensive collection of standard fonts. Unfortunately, you must become a wizard in order to print with your own new font. You can't use a new font unless it is added to the font dictionary on your printer, and adding fonts to dictionaries is a delicate operation: a sad state of affairs. If the Press file that you send to a printer asks for a font that the printer doesn't have, it will attempt a reasonable substitution, and, in the case of Spruce, tell you about the substitution on the break page of your listing. If you have chronic font difficulties of this sort, contact a wizard.

There is a new print file format under development, called Interpress. The print servers that are part of the Star product speak a dialect of Interpress. A print file in Interpress format is called a *master*. Our local plans for printing Interpress masters involve converting them first into a printer-dependent print file in so-called PD format (with conventional extension ''.pd''). From there, a relatively simple driver program on each printer should be able to produce the final output.

The rest of this section was contributed by Julian Orr of ISL:

PARC has a variety of printers available for your hardcopy needs. We have high volume printers for quantities of text, listings, and documentation; we have slower printers with generally higher quality for more complex files; and we have very slow printers for extremely high quality. All of our current printers except Platemaker offer 384 spots per inch and share a common font dictionary. We use two different software systems for printing Press files, both running on Altos: one is called Spruce, and the other is called (confusingly) Press. Spruce offers speed and spooling, but it can only image characters and rules, and not too many of them. This makes it limited in graphics applications. Furthermore, Spruce is limited to the particular sizes of fonts that it has stored in its font dictionary: it does not know how to build new sizes by converting from splines. Press is slower, but can handle arbitrary bitmaps, and can produce odd-sized fonts from splines.

ISL is developing Interpress printing capabilities. Printing ''.pd'' files is now an option on most Press printers (that is, on printers running the program Press as opposed to Spruce). Just ship your ''.pd'' file to the printer in the standard way: it is smart enough to figure out whether what you have sent it is in PD or Press format, and it will invoke PDPrint or Press as appropriate. Documentation on these two printing programs is available, by the way <31, 32>. PD printing should not be undertaken without consultation with a wizard.

Dover printers run Spruce for high volume printing, producing a page per second. CSL's Dover, named Clover, is found in room 2106; ISL's Dover, named Menlo, is in room 2305. Samples of the Dover font dictionary may be found next to Clover and Menlo. Instructions for modifying the queue and generally running these Spruce printers are to be found next to their Alto terminals.

Lilac is our color Press printer and may be found in 2106 with Clover. It is a three color, composite-black machine; it generally produces good quality output, but is occasionally temperamental. Anyone interested in color printing or the state of Lilac should join the distribution list LilacLovers^.pa.

In the ISL maze area, room 2301, we have an assortment of black and white Press printers, answering variously to the names of RockNRoll, Quoth, and Stinger. The printers are two Ravens (Raven is a Xerox product), one Hornet, and one Gnat (the latter two are prototypes). The print quality is normally excellent. Instructions for interpreting status displays are posted locally. To be informed of which printer is functioning and where, join the list ISLPrint^.pa. There should be three printers up for most of the summer. Periodically one or another of these or Lilac are pre-empted for debugging.

Our best quality printer is Platemaker, which is normally operated at 880 spots per inch, but can be run up to 2200 spi; it is not normally useful to go beyond 1600. Platemaker uses a laser to write on photographic paper or film. Color images can be done in individual separations, which are then merged using the Chromalin process. The Platemaker printing process is used for final prints of fine images or for printing masters for publication. If you wish to have something printed, speak to Julian Orr, Eric Larson, or Gary Starkweather.

Sending and Receiving Mail

We rely very heavily on an electronic mail system. We use it for mail and also for the type of announcement that might, in other environments, be posted on a physical or electronic bulletin board. In our environment, a physical bulletin board is pretty useless, since people spend too much of their days staring at their terminals and too little wandering the halls. Electronic bulletin boards might work satisfactorily. But a bulletin board, being a shared file to which many people have write access, is a rather tricky thing in a distributed environment. It probably presupposes a distributed transactional file server, for example. Mumble. For whatever reason, the fact remains that we don't have an electronic bulletin board facility at the moment. As a result, announcements of impending meetings, ''for sale'' notices, and the like are all sent as messages directed at expansive distribution lists. If you don't check your messages once a day or so, you will soon find yourself out of touch (and saddled with a mailbox full of obsolete junk mail). And conversely, if you don't make moves to get on the right distribution lists early, you may miss lots of interesting mail. This business of using the message system for rapid distribution of announcements can get out of hand. One occasionally receives notices of the form: ''meeting X will start in 2 minutes all interested parties please attend''.

Grapevine is the distributed transport mechanism that delivers the local mail [33]. When talking to Grapevine, individuals are referred to by a two-part name called an ''R-name'', which consists of a prefix and a registry separated by a dot; for example, ''Ramshaw.pa'' means Ramshaw of Palo Alto. In addition to delivering the mail, Grapevine also maintains a distributed database of distribution lists. A distribution list is also referred to by an R-name, whose prefix conventionally ends in the character up-arrow, as in ''CSL^.pa''. Distribution lists are actually special cases of a construct called a Grapevine ''group''. Groups can be used for such purposes as controlling access to IFS directories. There is a program named Maintain that allows you to query and update the state of the distribution list database. In fact, there are two versions of Maintain: the documented one with the unfortunate teletype-style user interface is used from within Laurel or the Mesa Development Environment <34>; the undocumented one with the futuristic menu interface is used from within Cedar. Some distribution lists are set up so that you may add or remove yourself using Maintain. If you try to add yourself to Foo^.pa and Maintain won't let you, the proper recourse is to send a message to the distribution list Owners-Foo^.pa, asking that you please be added to Foo^.

At the moment, Grapevine pretty much has a monopoly on delivering the mail. But there are several different programs that give users access to Grapevine's facilities from different environments. From an Alto, one uses Laurel, which is mentioned elsewhere as a pioneer of modeless editor interfaces. Even if you aren't a Laurel user, I recommend that you read Chapter 6 of the Laurel Manual [35], which is an enlightening and entertaining essay on proper manners in the use of the mail system. In the Mesa Development Environment, the program Hardy provides services analogous to Laurel's. From within Cedar, most folk use Walnut, whose documentation appears as part of the Cedar Manual <25>. Walnut represents a step towards the future in some respects, since Walnut uses Cypress, the Cedar database management system, to store your mail in a database. Access to Grapevine from within Cedar can also be had without the database frills through a program called Peanut, which stores your messages in a structured Tioga document instead of in a database. Finally, in case travel should take you away from your multi-function personal workstation, there are servers on the Internet known by the name ''Lily'' to whom you can connect from any random teletype in order to peruse the mail sitting in your Grapevine mailbox.

Packaging Systems and Controlling Versions

In the BCPL world, the primary facility for packaging up a group of related files and either handing them out or storing them for later recall was the *dump file*. A dump file, given conventional extension ''.dm'', is simply the concatenation of the dumpees, together with enough header information to allow the dumpees to be pulled apart again. Dump files have fallen out of favor.

In the Alto/Mesa world, and more strongly, in the Cedar world, a collection of software called ''DF files'' has grown up that attacks the problem of describing and packaging systems, detailing their interdependencies, and controlling the versions of things. You can find out a lot about DF files by reading Eric Schmidt's dissertation [36]. You can find the answers to detailed questions about the behavior of the various programs that deal with DF files by reading the reference manual for DF files <37>. All that I will try to do here is to give you some idea of what DF files are good for, and how, in a general sense, they are used. One way or another, all Cedar programmers must make their peace with DF files: they perform valuable functions, and they have no current competition.

In the simplest case, a DF file just consists of a list of the names of a related set of files. At this level, a DF file is something like a dump file: given the DF file, you can get at each of the files that it describes. Of course, you want to be sure that you get the right versions of the described files, so just having the DF file list their names isn't quite enough. If there were an Internet-wide notion of version number that made sense, we could get around this problem by specifying the version number along with the file name. But there isn't. The closest thing to a Internet-wide unique identification stamp that we have is the create date of the file. Thus, what a DF file really contains is a list of file names and associated create dates.

The first program that you will meet that deals with DF files is Bringover. Bringover's job is to retrieve to your local file system the set of files described by a particular DF file. This would be something of a challenge unless the DF file included some hint to Bringover concerning where in the great, wide Internet the correct versions of these files might be found. So DF files do indeed include such hints: in particular, they include specifications of remote directories on which to look. These directories are just hints, in the sense that Bringover will always verify by checking the create date that it is getting you the correct version of the specified file. If Bringover can't find the correct version on the specified directory, it will issue a sprightly error message. Bringover has lots of bells and whistles. For example, you can point it at either a local or a remote DF file; you can ask it to retrieve just a selected subset of files to your local disk, rather than the entire set described by the DF file; or you can individually consider the files one by one, deciding which you would like to retrieve and which you wouldn't.

Suppose that I am working on a collection of files, such as the sources for this Briefing Blurb. I have made a DF file that describes them, and I can use Bringover to retrieve them from their remote and permanent home to my local file system, where I can edit them. The next thing that I need is a service that is symmetric to Bringover: after doing my editing, I want to put the new versions back on the remote file server, along with a new DF file that describes the new versions. This function is performed by SModel. I run SModel, and point it at the old DF file. SModel considers each file in turn, and looks to see if I have edited it; that is, it looks to see if the create date of that file in my local file system is now different than the create date stored in the old DF file. If so, SModel deduces that I have edited the file. It stores the new version that I have made out onto the remote directory. After doing this for each file in turn, SModel writes a new version of the DF file itself, filling in all of the create dates correctly to describe the new version of the entire ensemble. If the DF file describes itself, as most DF files do, SModel is smart enough to make sure that the new version of the DF file is stored out to the remote server as well. SModel also has lots of bells and whistles, but let's not go into them.

If that were the whole story, mere mortals could figure out DF files without straining their brains. But there's more. So far, we have only discussed DF files as descriptions of ensembles of files. In fact,

these ensembles are often components of large programs. And this has consequences.

First, there are two distinctly different reasons that you might have for retrieving a program: you might want to change it, or you might just want to run it. In the latter case, you don't need to bring over all of the sources; all that you need is the runnable ''.bcd''. We could handle this by having two DF files: one for the users and the other for the maintainers. But that would be a disaster: the two DF files would never agree! Instead, each DF file distinguishes between files that it ''exports'' and the rest. The exported files are the ones that users need, while maintainers are assumed to need the entire ensemble. You can warn Bringover that you are a user rather than a maintainer by giving it the ''/p'' switch (which stands for Public-only).

Secondly, some programs are going to depend upon other programs: that is, the programs themselves will be ''users'' (''clients'' is a better word here). This suggests that one DF file should be able to contain another DF file. In fact, there should be several different kinds of containment, corresponding to such phrases as:

''They who maintain me also maintain the stuff described by the following DF file.''

''They who maintain me are also users (but not maintainers) of the stuff described . . .''

''They who use me are also users of the stuff described . . .''

You get the point? For more details on the ways that these things are done (''includes'' and ''imports''), check out the reference manual.

In case you still aren't convinced that things are complicated, it is now time to mention the fact that DF files are used for yet another purpose: they describe components of the Cedar release. During the Cedar release process, all of the new versions of Cedar components, which are sitting out on development directories, must be checked for consistency, and then moved en masse to the official release directory. And an entire new set of DF files must be produced, describing the released version of the system (as opposed to the development version). This means, among other things, that some DF files must specify two different remote directories: the development directory and the release directory. In addition, there is a third DF file program, called VerifyDF, whose purpose is to perform certain consistency and completeness checks on a DF file. By insisting that all component implementers have successfully run VerifyDF on their components, the Cedar Release Master ensures that the release process has at least a fighting chance of succeeding <38>.

In fact, there are several other programs related to DF files that are sometimes useful. DFDisk, for example, looks at all of the files on your local disk and classifies them according to where they may be found on remote servers. This is a convenient way to determine what local files need to be backed up before erasing the local file system for some reason. For more on DFDisk, an introduction to DFDelete, and more, see the DF files reference manual <37>.

DF files grew up over time in response to a mixed bag of needs. As they became more popular, features were added one by one to make them more useful in these varying contexts. The resulting system as a whole is rather hard to grok, but I hope that this introduction has given you a leg up on the problem, at least.

Some Tidbits of Lore

About CSL

CSL has a weekly meeting on Wednesday afternoons called Dealer, starting at 1:15. The name comes from the concept of ''dealer's choice'' the dealer sets the ground rules and topic(s) for discussion. When someone says she will ''give a Dealer on X'', she means that she will discuss X at some future weekly meeting, taking about 15 minutes to do so (plus whatever discussion is generated). Generally, such discussions are informal, and presentations of half-baked ideas are encouraged. The topic under discussion may be long-range, ill-formed, controversial, or all of the above. Comments from the audience are encouraged, indeed, provoked. More formal presentations occur at the Computer Forum on Thursday afternoons; the Forum is not specifically a CSL function, and it is open to all Xerox employees, and sometimes also to outsiders. Dealers are also used for announcements that are not appropriate for distribution by electronic mail. Members of CSL are expected to make a serious effort to attend Dealer.

On occasions of great festivity, Dealer is replaced by a picnic on the hill (that is, Coyote Hill, across Coyote Hill Road), with Mother Xerox picking up the tab.

The CSL Archives (not to be confused with TENEX archiving) are a collection of file cabinets and 3-ring binders that provide a continuing record of CSL technical activities. The archives are our primary line of defense in legal matters pertaining to our projects. They also make interesting reading for anyone curious about the history of any particular project.

There is also an institution known as the CSL Notebook, which exists to make all of the potentially interesting documentary output of CSL folk easily accessible to all CSL folk. Trip reports, design documents, immigration manuals (like this one): they should all be submitted to the CSL Notebook <39>. If you thought that it was worth writing down, it is pretty likely that there are other folk in CSL who would consider it worth reading, and submitting it to the CSL Notebook is one easy way to get it read. (I believe that likely looking submissions to the CSL Notebook are considered for entry into the CSL Archives as well.)

About ISL

ISL also has a weekly meeting, on Tuesdays starting at 11:00 am. This meeting has no catchy name at the moment.

Some Code Phrases

You may occasionally hear the following incomprehensible phrases used in discussions, sometimes accompanied by laughter. To keep you from feeling left out, we offer the following translations:

''Committing error 33''

(1) Predicating one research effort upon the success of another. (2) Allowing your own research effort to be placed on the critical path of some other project (be it a research effort or not). Known elsewhere as Forgie's principle.

''You can tell the pioneers by the arrows in their backs.''

Essentially self-explanatory. Usually applied to the bold souls who attempt to use brand-new software systems, or to use older software systems in clever, novel, and therefore unanticipated ways ... with predictable consequences. Also heard with ''asses''replacing ''backs''.

''We're having a printing discussion.''

Refers to a protracted, low-level, time-consuming, generally pointless discussion of something peripherally interesting to all. Historically, printing discussions were of far greater importance than they are now. You can see why when you consider that printing was once done by carrying magnetic tapes from Maxc to a Nova that ran an XGP.

Fontology

The body of knowledge dealing with the construction and use of new fonts. It has been said that fontology recapitulates file-ogeny.

''What you see is what you get.''

Used specifically in reference to the treatment of visual images by various systems, e.g., a Bravo screen display should be as close as possible to the hardcopy version of the same text. Also known is some circles by the acronym ''WYSIWYG'', pronuonced ''whiz-ee-wig''.

''Moving right along'', *''Pop!''*, or *''Hey guys, up-level!''*

Each of these phrases means that the conversation has degenerated in some respect, often by becoming enmeshed in nitty-gritty details. Feel free to shout out one or more of these phrases if you feel that a printing discussion has been going on long enough. If two participants in a large meeting begin discussing details that are of interest to them but not of interest to the group as a whole, shout *''Off-line!''* instead.

''Life is hard''

Two possible interpretations: (1) ''While your suggestion may have some merit, I will behave as though I hadn't heard it.'' (2) ''While your suggestion has obvious merit, equally obvious circumstances prevent it from being seriously considered.'' The charm of this phrase lies precisely in this subtle but important ambiguity.

''What's a spline?''

''You have just used a term that I've heard for a year and a half, and I feel I should know, but don't. My curiosity has finally overcome my guilt.'' Moral: don't hesitate to ask questions, even if they seem obvious.

Hints for Gracious Living

There are a couple of areas where life at PARC can be made more pleasant if everyone is polite and thoughtful enough to go to some effort to help out. Here are a few words to the wise:

Coffee

Both ISL and CSL have coffee alcoves where tea, cocoa, and several kinds of coffee are available. All coffee drinkers (not just the secretaries or some other such barbarism) help out by making coffee. If you are about to consume enough coffee that you would leave less than a full cup in the pot, it is your responsibility to make a fresh pot, following the posted instructions. There are lots of coffee fanatics around, and they get irritated beyond all reason if the coffee situation isn't working out smoothly. For those coffees for which beans are freshly ground, the local custom is to pipeline grinding and brewing. That is, you are expected to grind a cup of beans while brewing a pot of coffee from the previous load of ground beans. This speeds up the brewing process for everyone, since a load of ground beans is at least, had better be always ready when the coffee pot runs out.

Sharing Office Space

Be warned as well that some lab members are unbelievably picky about the state of their offices. The convention is that any Alto in an empty office is fair game to be borrowed. Private Dolphins and Dorados may be borrowed only by prior arrangement with their owners, because of the problems of sharing disk space. If you use someone's office for any reason, take care to put everything back *exactly* the way it was. Don't spill crumbs around, or leave your half-empty cocoa cup on the desk, or forget to put the machine back in the state that you found it, or whatever. Of course, lots of people wouldn't mind even if you were less than fanatically careful. But some people do mind, and there is no point in irritating people unnecessarily.

Sharing printers

When you pick up your output from a printer, it is considered antisocial merely to lift your pages off the top of the output hopper, and leave the rest there. Take a moment to sort the output into the labelled bins. Sorting output is the responsibility of everyone who prints, just as making coffee is the responsibility of everyone who drinks (coffee). Check carefully to make sure that you catch every break page: short outputs have a way of going unnoticed, and hence being missorted, especially when they come out right next to a long output in the stack. The rule for determining which bin is to use the first letter that appears in the name on the break page. Thus, ''Ramshaw, Lyle'' should be sorted under ''R'', while ''Lyle Ramshaw'' should be sorted under ''L''. A trickier question is what to do with output for ''Noname'', or the like. Following the rule would suggest filing such output under ''N'',but that doesn't seem very helpful, since the originator probably won't find it. Check the contents and file it in the right box if you happen to recognize whose output it is; otherwise, either leave it on top of the printer or stick it back in the output hopper.

The phone system

When the Voice Project has had its way, our phone system will be a marvelous assemblage of computers talking to computers, and this section of the Briefing Blurb will have to be expanded to tell you all about it. At the moment, however, we are simply customers of Pacific Telephone, so there isn't too much to say. First, a little preaching.

If you make a significant number of personal long-distance phone calls from Xerox phones, it is your responsibility to arrange to reimburse Xerox for them. This may not be that easy, either, since phone bills take quite a while (six weeks or so) to percolate through the bureaucracy upstairs, and the said bureaucracy also has a lot of trouble figuring out where to send the phone bills of new people, and people who move around a lot. Just because it is easy to steal phone service from Xerox doesn't make it morally right; if you think you aren't being paid enough, you should start agitating for a raise. If enough suspicious calls are made without restitution, PARC (being a bureaucracy) will impose some bureaucratic ''solution'' on all of us.

So as not to end on a sour note, let's discuss how the phone system works, anyway. The offices within PARC have four-digit extensions within the 494 exchange, a system known as Centrex; to dial another office, those four digits suffice. Dialing a single 9 as the first digit gives you an outside line, and you are now a normal customer of Ma Bell: see a phone book for more details (Oh, come now, surely you know about phone books!). Dialing a single 8 gives you different sounding dial tone, and puts you onto the IntelNet (not to be confused with the InterNet). The IntelNet is a Xerox-wide company phone system, complete with its own phone book, and its own phone numbers. If you are calling someone in some remote part of Xerox, you can save Mother Xerox some bread by using the IntelNet instead of going straight out over Ma Bell's lines. On the other hand, you may not get as good a circuit to talk over although this situation is frequently said to be improving. Furthermore, through the wonders of modern electronics, you can dial any long-distance number over the IntelNet. Just use the normal area code and Ma Bell number: the circuitry is smart enough to take you as far as possible towards your destination along IntelNet wires, and then switch you over to Ma Bell lines for the rest of the trip. Using the IntelNet doesn't start to save money until the call is going a fair distance; therefore, the IntelNet doesn't let you call outside numbers in area codes 408, 415, and 916 better to just dial 9.

One more thing: after you have dialed a number on the IntelNet, you will hear a funny little beeping. At that point, you are being asked to key in a four-digit number to which the call should be billed. You should use the four-digit extension number for your normal office phone under most circumstances. Calls made by dialing 9 instead of 8 are always charged to the phone from which they are placed.

The first three rings (roughly speaking) of an incoming call occur only in your office. The next roughly three rings happen both at your office phone and at a receptionist's phone, centrally located in the laboratory. During normal business hours, the receptionist's phones are staffed; thus, someone will at least take a message for you, and leave it on a little slip of paper in your physical message box. If the second three rings go by without either of those two phones answering, the call is then forwarded to the guards desk downstairs (I believe).

If you are expecting a call but won't be near your normal phone, a call forwarding facility exists: dial 106 and then the number to which you want your calls to be forwarded. Later on (*try* not to forget), you dial 107 on your normal phone to cancel the forwarding. When I forward my phone, I turn the phone around physically, so that the touch-pad faces the wall. This helps me to remember to cancel the forwarding again later, at which point I turn the phone back the normal way. There is also a way to transfer incoming calls to a different Xerox number: Depress the switch hook once, and dial the destination number; when the destination answers, you will be talking to the destination but the original caller won't be able to hear your conversation; depressing the switch hook again puts all three of you on the line; then you can hang up when you please. If the destination doesn't answer, depressing the switch hook once again will flush the annoying ringing or busy signal.

References

Reference numbers in [square brackets] are for conventional hardcopy documents. Many of them are Xerox reports published in blue and white covers; the CSL blue-and-whites are available on bookshelves in the CSL Alcove. Reference numbers in <angle brackets> are for on-line documents. The path name for such files is given herein in the form

[FileServer]<Directory>SubDirectory>FileName.Extension

for backward compatibility with earlier systems. Recently, the simpler alternative form

/FileServer/Directory/SubDirectory/FileName.Extension

has begun to come into local currency, but some systems still demand brackets rather than slashes.

<*n*>: The generic form for a reference to an on-line document.

[*n*]: The generic form for a reference to a hardcopy document.

[1]: **Sunset New Western Garden Book.** Lane Publishing Company, Menlo Park, CA, 1979. The definitive document on Western gardening for non-botanists; 1200 plant identification drawings; comprehensive Western plant encyclopedia; zoned for all Western climates; plant selection guide in color.

[2]: John E. Warnock. **The Display of Characters Using Gray Level Sample Arrays** blue-and-white report CSL-80-6.

[3]: Richard F. Lyon. **The Optical Mouse, and an Architectural Methodology for Smart Digital Sensors**. blue-and-white report VLSI-81-1.

[4]: **The Ethernet Local Network: Three Reports**. blue-and-white report CSL-80-2.

[5]: John F. Shoch, Yogen K. Dalal, Ronald C. Crane, and David D. Redell. **Evolution of the Ethernet Local Computer Network**. blue-and-white report OPD-T8102.

<6>: [Maxc]<AltoDocs>NetTopology.press. Contains a picture of the entire internetwork configuration in seven pages. It is out of date. All such documents are always out of date. A copy is posted on the wall opposite the Alcove in CSL.

[7]: David R. Boggs, John F. Shoch, Edward A. Taft, and Robert M. Metcalfe. **Pup: An Internetwork Architecture**. blue-and-white report CSL-79-10.

[8]: **Internet Transport Protocols**. Xerox System Integration Standard report XSIS 028112, December 1981.

[9]: **Courier: The Remote Procedure Call Protocol**. Xerox System Integration Standard report XSIS 038112, December 1981.

[10]: C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs. **Alto: A personal computer**. blue-and-white report CSL-79-11.

<11>: [Maxc]<AltoDocs>AltoHardware.press. Everything that you need to know to write your own Alto microcode.

[12]: **The Dorado: A High-Performance Personal Computer; Three Papers.** blue-and-white report CSL-81-1.

<13>: [Indigo]<DoradoDocs>DoradoBooting.press. Describes how to boot a Dorado, and how to configure it to boot in various ways.

[14]: Myer, T. H. and Barnaby, J. R. **TENEX Executive Language Manual for Users.** Available from Arpa Network Information Center as NIC 16874, but in the relatively unlikely event that you need one, borrow one from a Tenex wizard.

<15>: [Maxc]<AltoDocs>BCPL.press. The reference manual for the BCPL programming language.

<16>: [Maxc]<AltoDocs>OS.press. The programmer's reference manual for the Alto Operating System, including detailed information on the services provided and the interface requirements.

<17>: [Maxc]<AltoDocs>Packages.press. A catalogue giving documentation for the various BCPL packages that other hacker's have made available.

[18]: James G. Mitchell, William Maybury, and Richard Sweet. **Mesa Language Manual, Version 5.0.** blue-and-white report CSL-79-3. A cross between a tutorial and a reference manual, though much closer to the latter than the former.

<19>: [Ivy]<Mesa>Doc>Compiler60.press. Describes the changes in the Mesa language and the compiler that occurred in moving from Mesa 5.0 to Mesa 6.0.

[20]: Morris, J. H. **The Elements of Mesa Style.** Xerox PARC Internal Report, June 1976. Somewhat out of date (since Mesa has changed under it), but a readable introduction to some useful program structuring techniques in Mesa.

[21]: Adele Goldberg and David Robson. **Smalltalk-80: The Language and Its Implementation**. book published by Addison-Wesley, 1983.

[22]: Warren Teitelman. **Interlisp Reference Manual**. Published in a blue and white cover, although not officially a blue-and-white. October, 1978.

[23]: The Interlisp-D Group. **Papers on Interlisp-D**. blue-and-white report CIS-5 (also given the number SSL-80-4), Revised version, July 1981.

[24]: L. Peter Deutsch and Edward A. Taft, editors. **Requirements for an Experimental Programming Environment** blue-and-white report CSL-80-10.

<25>: [Indigo]<Cedar>Documentation>Manual.df. Hardcopies are entitled **The Cedar Manual**.

[26]: **Alto User's Handbook**. Internal report, published in a black cover. The version of September, 1979 is identical to the version of November, 1978 except for the date on the cover and title page. Includes sections on Bravo, Laurel, FTP, Draw, Markup, and Neptune

<27>: [Maxc]<AltoDocs>SubSystems.press. Documentation on individual Alto subsystems, collected in a single file. Individual systems are documented on [Maxc]<AltoDocs>systemname.TTY, and these files are sometimes more recent than SubSystems.press.

[28]: Jerome, Suzan. **Bravo Course Outline.** Internal report, published in a red cover. Oriented to non-programmers.

<29>: [Indigo]<Tioga>Documentation>TiogaDoc.tioga, or TiogaDoc.press. How to use the Tioga editor.

<30>: [Maxc]<PrintingDocs>PressFormat.press. Describes the Press print file format.

<31>: [Maxc]<PrintingDocs>PressOps.press. Describes the Press printing program.

<32>: [Maxc]<PrintingDocs>PDPrintOps.press. Describes the PDPrint printing program.

[33]: Andrew D. Birrell, Roy Levin, Roger M. Needham, and Michael D. Schroeder. **Grapevine: an Exercise in Distributed Computing.** blue-and-white report CSL-82-4.

<34>: [Ivy]<Laurel>Maintain.press. Documentation for the teletype version of Maintain, the version that is used from within Laurel or Tajo.

[35]: Douglas K. Brotz. **The Laurel Manual.** blue-and-white report CSL-81-6.

[36]: Eric Emerson Schmidt. **Controlling Large Software Development in a Distributed Environment.** blue-and-white report CSL-82-7.

<37>: [Indigo]<Cedar>Documentation>DFFilesRefMan.press. The reference manual for the use of DF files.

<38>: [Indigo]<Cedar>Documentation>ReleaseProcedures.press. Describes the policies and procedures that individuals who contribute to Cedar releases need to understand and observe.

<39>: [Indigo]<CSL-Notebook>Docs>HowToUseCSLNotebook.press.

The Briefing Blurb Glossary:

Terms, Acronyms, Directories, Files, and Protocols of either Current or Historical Interest

1983 Edition

By Lyle Ramshaw of PARC/CSL

Try reading me in Tioga, using the ‘‘Def’’ command to get around!

June 7, 1983

Filed on: [Indigo]<Cedar>Documentation>Glossary.tioga, Glossary.press

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

For Internal Use Only

abstract machine: A set of low-level functions and capabilities, provided by some combination of hardware and software, that forms the underpinnings of a system sitting above. For example, the **Interlisp-D** system, which runs on various machines, consists of a lot of machine-independent stuff sitting on top of a small amount of machine-dependent code. The goals of the machine-dependent part were specified by describing an abstract machine that it must implement. As another example, part of the **Cedar** project has been the specification of a ''Cedar computer'' as an abstract machine.

Alcove: See **CSL Alcove**.

Alpine: A transactional file server being built within **CSL** on top of **Cedar** for use by database systems and other distributed computing applications. Some folk are now storing their **Walnut** mail databases on Alpine.

Alto: (On its way to being archaic.) A small personal computer with a **bitmap** display and **mouse**, designed at **PARC**; the precursor to **D-machine**s. See the **blue-and-white** report titled ''Alto: A personal Computer'', number CSL-79-11.

Alto world: An environment created by running an **Alto emulator** on a **D-machine**.

AltoFontGuide.Press: A file, available on **[Indigo]<Fonts>**, that tells all about the existing families of display-screen raster **font**s, and describes how they are organized on different subdirectories of **[Indigo]<AltoFonts>**. Note that the name ''AltoFonts'' is an anachronism, and should really be changed to ''DisplayFonts'', or ''ScreenFonts'', or the like; the same rasters that were drawn for use on **Alto**s work just fine on today's **D-machine**s.

AM: Acronym for the **Cedar** abstract machine.

ARPA: Acronym for the Advanced Research Projects Agency of the United States Department of Defense. They support, among other things, a network linking research computers: our ARPANET address is PARC-MAXC.

atom: (or ATOM:) Unique identifiers implemented over a global naming space. Two occurrences of the same atom will evaluate to the identical value, rather than just to equivalent values. Atoms have always been part of **Interlisp**; they were added to **Mesa** on the way to **Cedar**. In **Cedar**, an atom literal is written with a prefixed dollar sign, as in ''$foo''. Each atom has a list of <name, value> pairs associated with it, called its *property list*.

Auditorium: A lecture hall in the new wing of **Building 35**.

bank: A unit of measurement of primary storage in an **Alto world**, equal to 64K 16-bits words, that is, 128K bytes. An **Alto** II has four banks, while **Dorado**s have at least eight.

bar: A generally thin, generally rectangular, generally invisible region of the screen in which certain generally display-related actions occur, e.g., the scroll bar, the line-select bar.

baseboard: A microcomputer that lives on the lowest printed-circuit board of a **Dorado**. The baseboard listens to the terminal's **boot button**, and to various thermometers. Its job is to supervise the rather complex booting sequence necessary for bringing a Dorado up from a cold start. The baseboard announces its state to the outside world by flashing a number (encoded in unary) on a little green light on the Dorado chassis. Signs near each bank of Dorados explain what various numbers of flashes mean.

Bayhill: Another name for **Building 96,** occupied by part of **SDD**. The **Bayhill** building is located on Hillview just before it runs into Arastradero.

BCD: A compiled object program module in **Mesa** or **Cedar**; an acronym for Binary Configuration Description.

BCPL: A free-wheeling and typeless system programming language used as the environment for much early Alto programming. Also, the compiler for that language.

BFS: An acronym for Basic File System; the contents of a disk or **partition** used by an **Alto world**. Also a standard software package for low-level management of an Alto file system.

Binder: BCD's **export** services to their **client**s, and, in turn, **import** various services from other

BCD's. The process of resolving these inter-module references is called *binding*, and the Binder is the program that does it. Actually, the loader can handle many of the easy cases of binding on the fly, as part of the loading process; but for complex stuff, you need the Binder. The Binder accepts compiled modules (with extension ''.bcd'')and binding instructions in the form of a configuration description (with extension ''.config''); it produces another ''.bcd'' as output.

BITBLT: (pronounced ''bit-blit''). A complex instruction used for moving and possibly modifying a rectangular **bitmap**. The ''BLT'' part is an acronym for BLock Transfer.

bitmap: Generally refers to a representation of a graphical entity as a sequence of bits directly representing image intensity at the points of a raster. The display hardware and microcode on an **Alto** or **D-machine** process what is essentially a bitmap of the image to be displayed. At PARC, bitmaps are normally stored word-aligned, and in row-major order.

blue-and-white: A report that has been cleared for distribution outside Xerox, and published in a blue and white cover. Such reports have identifying numbers formed by concatenating the laboratory acronym, the year, and a small integer. One of my favorites is the Laurel Manual, by Douglas K. Brotz, number CSL-81-6; I especially recommend Chapter 6. **CSL** blue-and-whites are stored on bookshelves in the CSL Alcove. A list giving the titles and numbers of all of the blue-and-whites is available from the PARC Library.

Bluejay: The **Etherphone** voice file server.

boot: Short for ''bootstrap'', which is in turn short for ''bootstrap load''. Refers to the process of loading and starting a program on a machine whose main memory has undefined contents.

boot button: The small button behind the keyboard used (sometimes in conjunction with the keyboard) to **boot** some program into execution. On **Dolphin's** or **Dorado's**, there are other more potent boot buttons on the chassis, in addition to the boot button behind the keyboard.

boot file: A file that contains a bootable program. Used to start **Cedar**, as well as various games and other useful programs available from the **NetExec** in the **Alto world.**

boot server: A computer on the network that provides a retrieval service for certain stand-alone programs (which are encapsulated as **boot file**s). See **NetExec**.

Bravo: (archaic) An integrated text editor and document formatting program that runs on the **Alto**.

BravoBug: (archaic) A program used when Bravo crashes to **replay** the editing actions up to the point of the crash.

BravoX: A successor to **Bravo** written in **Butte** with somewhat greater functionality and a somewhat richer interface. Warning!: BravoX source files are stored in a weird and wonderful format that almost NO programs other than BravoX can handle. Also, BravoX runs, at the moment, only on Alto II's and (perhaps?) Dolphins.

break page: A header page that divides one printed file from another in the output of a **Spruce** printer. If Spruce encountered any difficulties during the printing run, it will inform you of them on the break page.

Bringover: A program that retrieves files from remote file servers to one's local disk; Bringover reads ''.df'' files in order to figure out what versions of what files should be retrieved, and where in the great wide electronic world they might be found. Use of Bringover (confusing as it may be at the outset) is to be recommended over use of either **FTP** (in the **Alto world**) or the **FileTool** (in **Cedar**), since the version control and system-description features of ''.df'' files are very valuable.

bug: A computing term for a non-feature, something that is not as intended. Sometimes used in a different sense to refer to the act of pointing at something with the mouse, and then clicking a mouse button; but this usage is frowned upon by 100% of our Usage Panel namely me. I recommend using the verb ''click''instead in this context, since I think that ''bug'' is already an overloaded word. I have also seen the verb ''hit'' used to refer to this action; I consider

it an acceptable alternative to ''click''.

bug award: Refers to a occasional custom within **CSL** and **ISL** wherein those brave souls responsible for ferreting out the cruelest and most intricate bugs in critically important systems are rewarded for their efforts by being presented with a cute little bug-shaped sticker that they can then display on their office nameplate or elsewhere. A bug award is the moral equivalent of a gold star. If the sticker consists of a background from which a bug has been excised, then the award is an ''inverse bug award'', and serves to praise its recipient for producing code that is notably free of insect infestations.

BugBane: A package that implements the basic primitives necessary for high-level debugging in the **Cedar** world; the **UserExec** is a **client** of BugBane, and, in turn, provides debugging services to **users** of Cedar.

Building 32: A part of **OSD**, located on Hanover Street, north of Page Mill. Once called **PARC-place**, when it was occupied by parts of PARC.

Building 34: A part of **PARC**, located on Hillview, just across Coyote Hill from the **Building 35**, the home of the **ICL**.

Building 35: The main building of **PARC,** located at the intersection of Coyote Hill and Hillview. The site of the cafeteria.

Building 37: A part of **PARC**, located on Hanover Street, north of Page Mill, and just south of **Building 32.** The site of the **CSL Electronic Model Shop.**

Building 96: A part of **OSD**, located where Hillview runs into Arastradero; also called the **Bayhill** building. Current home of some parts of **SDD.**

Butte: A compiler for **BCPL** that outputs **Mesa**-style byte codes instead of Nova assembly code; also, the byte codes themselves, and the microcode that implements them.

button: A small area on the screen that reacts when clicked with the mouse. In **Viewers**, buttons are rectangular areas labelled with a word or phrase; they are organised into **menus**.

byte code: **Lisp, Mesa, Cedar, Smalltalk,** and **Butte** at **PARC** compile into directly executable languages that are stack oriented, and whose op codes are usually one byte long. Such an instruction is called a **byte code.** These **byte code**s are in turn interpreted by special microcode on each of our various machines.

Cabernet: A particular mail server that is part of the **Grapevine** distributed transport mechanism, located in the **CSL** machine room.

caret: A blinking pointer, indicating where keyboard characters will appear when typed.

catch phrase: A chunk of **Mesa** or **Cedar** code that is prepared to handle a certain type of exceptional condition. One way to think of a catch phrase is as the body of a procedure variable that is dynamically bound. Such procedures variables are called **signals**. If you suspect that an exceptional condition might arise, and you think that you know what to do if it does, you specify this response as a catch phrase; that is, you bind a procedure value to the signal, which is a procedure variable. If any procedure that you call notices that the condition has in fact arisen, it will notify the world by ''raising the signal'', which should be thought of as a procedure call to the catch phrase that you specified. (This method of explaining signals is a minor facet of the religion espoused in the **CLRM**.)

Cedar: A large project in CSL to build a programming environment for CSL's future applications. Also the name of that environment. Also the name of the programming language upon which it is built. The Cedar language is a variant of **Mesa** augmented by garbage collection, **atom**s, and run-time types. The design of the **Cedar** environment was strongly influenced by the programming environment and services available in **Interlisp** and **Smalltalk**. For a discussion of the goals of Cedar, see the **blue-and-white** report titled ''Requirements for an Experimental Programming Environment'', number CSL-80-10.

CedarGraphics: A subroutine package of graphic primitives, built within **ISL**, that forms an

important part of **Cedar**. Its design was heavily influenced by the results of experimental systems written in **JaM.** Soon to be replaced by the **Imager**.

Chardonnay: A **Grapevine** server.

Chat: A program that provides teletype-like ''interactive'' access to a remote computer on the network. Most programming environments include this capability in some form; both Alto and Cedar include programs actually named ''Chat''. Chat is mainly used to communicate with **Maxc** and **IFS** servers.

Checkpoint: A method used in **Cedar** to preserve the state of your computing world. Taking a Checkpoint involves preserving a shapshot of the current state of the virtual memory, but not of the file system. If, after taking a Checkpoint, something bad happens and your Cedar system gets **wedged**, the command **RollBack** will return you to the earlier clean state of your virtual memory; but changes to the file system made between the Checkpoint and the subsequent RollBack, such as storing edited versions of files, will not be undone.

Cheshire: A subsidiary of Xerox. They make a machine that binds stacks of paper into booklets by melting glue and letting it be absorbed by the edges of the paper. There are Cheshire binders in **CSL** and in the PARC **TIC**.

Chromalin: The trade name of a fancy color printing process used with the **PlateMaker** for creating high-resolution color prints from **Press** files or **PD** files.

Chipmunk: A **D-machine Mesa** program for interactively creating and editing integrated circuit designs. Chipmunk makes use of a color display in addition to the normal black-and-white one. It is a successor to **Icarus.**

Cholla: A **Laurel**-based IC fabrication line control program, which is used in **ICL**.

CIFS: An acronym for Cedar Interim File System. CIFS is currently used within **Cedar** to manage a portion of the local disk as a cache containing readonly copies of remote files. This function and others will someday be provided by **FS**. CIFS was the first CSL system to allow the components of a hierarchical file name to be separated with simple slashes instead of with square brackets and angle brackets; the clumsier brackets are being used in this document (sigh) for compatibility with the past.

CIS: An acronym for Cognitive and Instructional Sciences Group. A part of PARC, and the home of many of the builders of **Interlisp-D**.

Clearinghouse: The analog of the **Grapevine** registration database in the **NS** world. That is, a machine running **Star** talks to the local Clearinghouse in order to find out how to talk to a particular **file server** or **print server**.

click: A manipulation of a mouse button. Pushing and releasing a mouse button several times in quick succession is sometimes called a ''double-click'', ''triple-click'', etc. as appropriate. The phrases ''click-hold'' and ''double-click-hold'' are also sometimes heard.

client: A program (as opposed to a person) that avails itself of the services of another program or system. **Laurel** is a **client** of **Grapevine.** See **user.**

Clover: A **Dover** located in **CSL**.

CloverFonts.Press: A file, available on **[Indigo]<Fonts>**, that lists by family name, face, size, and rotation all of the fonts in **Clover**'s font dictionary. That is, this file lists the fonts that you can print with; for the fonts that you can see on your screen, see **AltoFontGuide.Press** instead. To see the characters of the fonts in all their glory, check out the book located on top of Clover called CloverCharacters.Press.

CLRM: Acronym for the Cedar Language Reference Manual. This document isn't exactly easy bedtime reading, but it is the most authoritative description currently available of the behavior of Cedar programs in interesting and subtle cases. The CLRM also attempts to convert you to a particular religion regarding the proper design of a polymorphic language within the Algol tradition. To get the good dope about current Cedar without spending the time

necessary to undergo religious conversion, skip immediately to Chapters 3 and 4 of the CLRM.

CoCedar: A **world-swap** debugger for **Cedar.**

color display: A CRT display with red, green, and blue phosphors. **Griffin** and **Chipmunk** both use the color display, and the color display is also available to users of Cedar with a minimum of hassle through the good auspices of the Cedar **ColorDevice**. The public **Dorados** with color displays are listed at the sign-up sheets.

ColorDevice: A component of **Cedar** that provides low-level support for a **color display**.

Com.cm: A file used by the Alto **Executive** to store the current command being executed. See **Rem.cm**.

Commander: A ''light-weight'' command interpreter, providing the minimum of functionality needed by **Cedar** implementers while they are developing a new release of the system. Most users of Cedar can instead enjoy the more plentiful features of the **UserExec**.

component: Among many other things, a chunk of software that is distributed as part of a **Cedar release**.

config: A source file that tells the **Binder** how to assemble modules into a complete system.

CoPilot: A **world-swap** debugger for **Pilot.**

CopyDisk: A stand-alone program used to transfer an Alto **BFS**, that is, the entire contents of an Alto disk or **partition**. May be used between computers or on a single computer with multiple disk drives.

create date: When said of a file, the date and time that the information contained in this particular version of this particular file was created. Create dates are generally stored accurate to the nearest second. This makes them sufficiently unique that the pair <file name, file version's create date> can generally serve as a unique identifier for a particular pile of bits.

credentials: Proof that you are who you say you are; usually your **Grapevine R-name** and the corresponding password.

CSL: Acronym for Computer Science Laboratory, a part of **PARC**, located on the second floor of **Building 35**.

CSL Alcove: A small meeting area containing a large round table that is located near **Clover** in **CSL.**

CSL Notebook: A mechanism for distributing, indexing, and generally sharing the documentary output of folk in **CSL.**

CSS: Acronym for Computing Seminar Series: talks, often by visitors, on various computing topics, which are held in CSL on Tuesdays at 1:30.

cursor: A small picture on the display that tracks the motions of the **mouse**.

Cypress: A database package based upon an entity-value-relationship model of data, and written in **Cedar**. **Walnut**, **Hickory**, and **Squirrel** are clients of Cypress.

czar: A general term for a person who functions as a benevolent dictator over some area of local life; for example, we have a softball czar and a publications czar. In the latter case, the role of the czar is largely that of an editor.

D-machine: A generic name, referring to any of the current machines within Xerox that implement the **PrincOps** architecture: **Dandelions, Dicentras, Dolphins,** and **Dorados** are the primary **D-machine**s.

D0: (that is, ''D-zero'') An obsolete name for the **Dolphin**, a **D-machine**.

DA: Acronym for Design Automation, and the name of a project in **CSL** and **IDL** to produce tools that help people build hardware of various sorts.

Daffodil: An inexpensive **D-machine** using custom VLSI, being designed by local folk. Since the Daffodil may become the hardware base of future OSD products, certain details concerning

the Daffodil project are rather sensitive.

Daisy: A **Dover** located in the **Bayhill** building.

Dandelion: The name of the processor that is in the **Star** products; an example of a **D-machine**.

dead: Either not currently operational (said of a piece of hardware), or operational but not currently undergoing continued development and support (said of bodies of software).

Dealer: The name of **CSL**'s weekly meeting, occurring on Wednesday afternoons from 1:15 until 2:45 (or so); also used to refer to the person speaking at that meeting. Giving such a presentation is referred to as ''giving a Dealer'' or sometimes ''Dealing''. See also **weekly meeting.**

DDS: (archaic) Acronym for Descriptive Directory System. An Alto **subsystem** providing sophisticated manipulation of the Alto file directory system. See also **Neptune**.

DF files: A collection of programs for describing the files needed to build a complicated system, for automatically retrieving these files from remote **file server**s to the local disk (**Bringover**), and for storing them back later (**SModel**). Unlike the more grand and glorious **system models** to come, DF files primarily addresses the problems engendered by our current feudal collection of file systems. The letters ''DF'' are an acronym for Description Files, which suggests that the phrase ''DF files'' is redundant.

Dicentra: A recent and inexpensive **D-machine**. The Dicentra essentially consists of the **Dandelion**'s CPU minus the **tasking** stuff squeezed onto one **Multibus** card, and communicating with memory and with I/O device controllers over the Multibus.

dirtball: A small, perhaps struggling outsider; not in the major or even the minor leagues. For example, ''Xerox is not a **dirtball** company''.

distribution list: A list of **R-names** to which mail can be addressed. In some cases, **Maintain** can be used to add oneself to interesting **DL**'s, such as ''MesaFolklore^.pa''. If Maintain responds that you aren't allowed to do that, the correct recourse is to send a polite message to ''Owners-MesaFolklore^.pa'', asking that they please add you to their list. For more details about distribution lists, try either the **Laurel** manual or the document [Ivy]<Laurel>Maintain.Press, which describes the **Alto** and **Tajo** versions of Maintain.

DiskDescriptor: A file that contains the disk allocation information used by an Alto file system.

DL: Acronym for Distribution List**.**

DLS: Acronym for Data Line Scanner: an Alto equipped with lots of modems plus other hardware and microcode to allow dialing into and out of the **Internet**.

DMT.boot: Acronym for Display Memory Tester. A memory diagnostic for the **Alto world**. DMT is automatically booted from the network by the **Alto Executive** after the Alto has been idle for about 20 minutes. DMT accepts various commands; try pushing the "S" key, and also try typing shift-swat. Designing cursors for DMT is a popular sport: send your suggestion as a list of 16 octal numbers to David Boggs (Boggs.PA), along with a suggested title line and an indication of whether you want to be credited by name.

Dolphin: A **D-machine**; once called the **D0**. More flexible than a **Dandelion**, but also slower and more expensive.

Dorado: A high-performance **D-machine**, designed by **CSL** and coveted by all and sundry. See the **blue-and-white** report titled ''The Dorado: A High-Performance Personal Computer'', number CSL-81-1.

Dover: Generic name for a type of 384 bpi laser-scan printer built on the Xerox 7000 xerographic engine and connected to an Alto by means of a **Orbit** interface. Successor to **EARS. Dover**s are normally driven by the program **Spruce**.

Dragon: Generic name of a new, custom-chip processor being designed by a team in **CSL**; it is hoped that the **Dragon** will satisfy our ambitions to have ''a **Dorado** in a shoe box''.

Draw: (archaic) An Alto **subsystem** that permits interactive construction of pictures composed of lines, curves, and text. Draw users may be interested to note that a program ReDraw exists that converts Draw source files into **Press** files that will print without the **jaggies** on a **Dover.**

Users of **Alto emulators** on **D-machines** must use DDraw and ReDDraw instead.

Dumper.boot: (archaic) A file used for desperation debugging in an **Alto world**. Dumps (most of) the current core image to **Swatee** for subsequent inspection by a debugger.

DWIM: Acronym for Do What I Mean: a facility intended to help the programmer by making LISP do what you mean, rather than what you say.

EARS: (archaic) Acronym for Ether Alto Research **SLOT**. An obsolete prototype laser-scan printer built on the Xerox 7000 xerographic engine and equipped with a hardware character generator. (Interesting to some as an example of a third level acronym: the S in EARS stands for SLOT, and the L in SLOT stands for LASER, and LASER itself is an acronym!)

EditTool: A menu-oriented command interface to the **Tioga** editor, providing complete access to Tioga's functionality, including the commands that you can't type (either because they can't be typed, or because you have forgotten how to type them).

EFTP: A venerable **PUP**-based protocol now mostly used to transfer print files to **print server**s and **boot file**s to **Alto**s.

Electronic Model Shop: An arm of **CSL** located on Hanover street in **Building 37**; this group of folks do small-scale production runs of computer equipment for **CSL.** Frequently called the **Garage.**

EmPress: An Alto **subsystem** used to convert text files to **Press** format and ship them to a **Press print server**.

emulator: A technique in which one computer is programmed to imitate another. Fast imitations are called emulators, while sufficiently slow ones are called simulators. In particular, the microcode in an **Alto** or **D-machine** that implements either **Mesa**, **Lisp**, or **Smalltalk** is known as the emulator microcode.

EOS: Acronym for Electro-Optical Systems; an organization located in Pasadena that was formerly a part of Xerox. The defense contracting portion of EOS was recently sold by Xerox for 40 megabucks. The portion of EOS that built Scientific Information Systems is now **SIS**; they are the ones who are marketing **D-machine**s running **Interlisp** and **Smalltalk** to the outside world.

Ernestine: A particular **Lily** server located in Building 35.

Ethernet: The communication line connecting many computers (with compatible interfaces) together. Strictly speaking, an Ethernet is a single, continuous piece of co-axial cable, but the term is sometimes applied to the entire network accessible through the cooperation of **Gateway**s (which is more correctly called an **InterNet**). Ethernets come in two flavors: the original Ethernet, now called the Experimental Ethernet, was built within PARC and runs at 3 MBits/sec. The Ethernet that has been proposed as a communication standard is a re-engineering that runs at 10 MBits/sec. PARC currently has Ethernets of both these flavors running around, as well as a special 1.5MBits/sec Ethernet used by the **Etherphone**s. See the **blue-and-white** report titled ''The Ethernet Local Network: Three Reports'', number CSL-80-2.

Etherphone: A box of magic widgets that can replace your office telephone, giving you much greater functionality by taking advantage of the power of computing in general, and of your personal multi-function workstation in particular. An Etherphone has a microphone, a speaker, digital-to-analog and analog-to-digital converters, a connection to Ma Bell, an Ethernet interface, and several microprocessors to tie them all together. The Etherphone is a recent product of the **Voice** Project within **CSL**. The existence of the Etherphone should make it easy to write lots of exciting experimental systems (any volunteers to write a CedarVoice interface?).

Executive: A distinguished Alto **subsystem** that provides simple commands to inspect and manipulate the file system directory, and to initiate other subsystems.

export: A **Mesa** or **Cedar** program that provides (either some or all of) the services described in an **interface** is said to export that interface.

face: The interface that an I/O device presents to the **Pilot** operating system; for example, there is a disk face and a display face. Each face is codified as an **interface module**. Particular implementations of this interface for particular devices are called **head**s.

file extension: The portion of a file name that appears following a period (possibly null). By convention, a number of extensions are reserved to indicate the type of data in the file, though not all subsystems are consistent in their use of extensions. Some commonly encountered extensions are:

~ an Alto Executive command (not really an extension)

.al: screen font rasters in the original format

.bcd: Mesa object program module

.bcpl: BCPL source program module

.bfs: an entire Alto file system gathered into a file

.boot: program invokable by booting

.br: BCPL object program module

.bravo: text file containing Bravo formatting information

.cm: Executive command file

.config: Mesa source that describes how to combine modules

.df: description of a system for use with DF files software

.dl: distribution list (in a file as opposed to in Grapevine's database)

.dm: (archaic) dump file, i.e., several logical files stored as one

.errors: Swat error message file

.icons: file containing displayable Icon images

.image: executable Alto/Mesa program

.jam: JaM interpretable code

.ks: screen font rasters in a fancy format

.laurel: special flavor of .bcd that can be run within Laurel

.log: history of certain program actions

.mail: Laurel mail file

.mail-dmsTOC: Laurel table-of-contents file

.mesa: Mesa source program module

.pd: file in PD (=printer dependent) print file format, usually produced from an Interpress master

.press: print file in Press format

.profile: records a user's preferred values of various user interface parameters in Cedar

.run: executable Alto program, that is, a subsystem

.sil: SIL source file for a drawing

.st: Smalltalk source program text

.strike: screen font rasters in a compact and efficient but limited format

.style: Tioga document style rules for formatting

.symbols: Mesa symbol table (for debugging)

.syms: BCPL symbol table (for debugging)

.tex: TEX source text

.tfm: font metric information

.tip: TIP interaction description

.tioga: Tioga text document

file name: See **file extension** and **path name** for information about the local conventions for file names.

file server: A computer on the network that provides a file storage and retrieval service. **MAXC**, **IFS,** and **Alpine** are three different types of file servers.

FileTool: A program in Cedar that allows the user to store and retrieve files from and to remote file servers. Use of the FileTool to retrieve portions of large systems to one's local file system is fraught with peril, since it is quite important that one retrieve consistent versions of things if the large system is to work, and the FileTool doesn't include any scheme of version control. Cautious programmers use **Bringover** and ''**.df**'' files from the beginning; everyone uses **Bringover** and ''**.df**'' files eventually.

Finch: The program that runs in your workstation and helps to control your **Etherphone**.

FLG: (pronounced ''flug'') In LISP programs, a switch that customizes a program's behavior to an individual user's working habits.

fog index: A measure of prose obscurity. Units are years of education required in order to understand the measured prose.

font: An assortment of characters all of one size and style; more precisely, a mapping from a set of character code numbers to a consistent collection of graphic images.

Fonts.widths: A file containing character-width information for a large number of fonts. Used by some programs that do text formatting while producing **Press** files. The standard source is **[Indigo]<Fonts>Fonts.Widths.** Other programs appeal to separate ''.tfm'' files, one for each font, as their source of information about character metrics.

foo: The first meta-syntactic variable. The second is ''bar''. There is a tie for third between ''fum'' and ''baz''. The words ''foo'' and ''bar'' are cognates, both derived from ''fubar'', an acronym popular in the U.S. Navy and used by early computer programmers employed by the Navy, possibly as a technical term describing the state of a system.

Football: A two-person game in **Cedar**.

format: An attribute of a **node** in a **Tioga** document. Examples might be ''long quotation'', or ''item in a bulletted list''. The effect of the various formats is defined by the **style**.

Forum: A series of talks on topics of general interest to folk at **PARC**, held on Thursday afternoons at either 3:45 or 4:00 p.m. in the **Auditorium**.

FS: A file directory system that will emerge in **Cedar** along with the **Nucleus**; FS will replace **CIFS** and the Common Software Directory (a part of Pilot).

FTP: Acronym for File Transfer Protocol (or Program). An **Alto world** program that provides a convenient user interface to the file transfer protocol, enabling the transfer of files between co-operating computers on the **Internet**.

Garage: A nickname for the **Electronic Model Shop**, a part of **CSL**.

Gateway: A computer serving as a forwarding link between separate **Ethernet**s. Gateways may also perform certain server functions, such as **name lookup**.

germ: A small part of **Pilot** that runs first; the germ handles bootstrap loading, inloading and outloading memory images during **worldswap**s, **teledebug**ging, and the like.

Grapevine: The distributed electronic message transport system; it has a set of protocols all its own, and provides various server functions such as authentication. See the **blue-and-white** report titled ''Grapevine: an Exercise in Distributed Computing'', number CSL-82-4.

Griffin: A **Mesa** illustration program, a successor to **Draw**. Excellent on filled areas, and handles color. **Griffin** was the source of many of the pretty pictures hanging near **Lilac.**

group: (when referring to **Grapevine**) A set of **R-name**s. The standard interpretation of a group is a **distribution list**. For example, CSL^.PA is the group of all people in **CSL**, in case they all should get copies of a message. Groups can also be used for other purposes, such as access control. The R-names that constitute a group are called its *members*. In addition, a

group has *friends* and *owners:* a *friend* is someone who may add or delete herself from the group, while an *owner* may add or delete anyone from the group.

Hardy: A **Tool** that provides the functionality of **Laurel**, that is, mail sending and receiving, within **Tajo**; a client of **Grapevine.**

head: The program module within **Pilot** that controls a particular I/O device. For example, each particular model of disk drive has an associated head. All of these heads **export** the disk **face**.

Hickory: A reminder and calendar system based on the **Cypress** database in **Cedar.**

Hornet: Generic name for a family of 300 bpi laser-scanned printers, built on top of 2600 copiers.

Ibis: An IFS server in **SDD**/Palo Alto.

Icarus: (archiac) An Alto-based program for creating and editing integrated circuit designs graphically and interactively.

icon: A small image representing some concept. Used extensively in **Star** and **Cedar**.

Idun: An IFS server in **SDD**/Palo Alto: the home **file server** of the **Pilot** group.

ICL: Acronym for Integrated Circuit Laboratory, a part of **PARC**, located in **Building 34.**

IDL: Acronym for Integrated Design Laboratory, an incipient part of **PARC**. Once formed, IDL (to be pronounced ''ideal'' rather than ''idle'') will be located somewhere in **Building 35**.

IFS: Acronym for Interim File System. An Alto-based **file server**. Many IFS servers exist on various **Ethernet**s, including **Ivy**, **Indigo, Ibis**, **Iris**, **Idun**, **Igor**, **Phylum,** and Erie.

IFU: Acronym for Instruction Fetch Unit; many computers have them.

Igor: An IFS server in **SDD**/Palo Alto: the home **file server** of the **Mesa** group. This name should be pronounced ''Eye-gore'', as in the movie *Young Frankenstein*.

Imager: A new implementation of the **CedarGraphics** package that is under development.

implementation module: A **Mesa** or **Cedar** module that actually provides a set of services, as opposed to an **interface module**, which simply specifies exactly what those services are to be.

Indigo: An **IFS** server in **PARC**, used by **CSL** and **ISL** to store project software files.

[Indigo]<AltoDocs>: A directory on which documentation for various Alto subsystems are stored (generally with extension **.press**).

[Indigo]<AltoFonts>: A directory on which screen fonts for the Alto are stored (extensions **.al**, **.strike**, or **.ks**). Subdirectories are used on this directory to distinguish various families of display screen fonts that have accumulated over the years.

[Indigo]<BasicDisks>: A directory on which the standard starting configurations for Alto disks are stored, as files with extension ''**bfs**''. The normal way to initialize a new **Alto world** is to use **CopyDisk** to retrieve one of these disk images.

[Indigo]<Cedar>: A directory containing the **Cedar** source code and documentation.

[Indigo]<Cedar>Documentation>: A directory containing the on-line documentation for the latest version of **Cedar**.

[Indigo]<Cedar>Top>: A directory containing top level **.df** files for **components** of the current **Cedar release**.

[Indigo]<Fonts>: A directory containing various documents of printing interest, including **Fonts.widths**. You might be interested in **CloverFonts.Press**, or **AltoFontGuide.Press.**

[Indigo]<ISL>: A directory of packages released by **ISL** for use within **Cedar.** Contains mainly interactive graphics software and document formatting tools.

[Indigo]<PreCedar>: A development directory for **[Indigo]<Cedar>**; that is, this is where **components** of a new release of **Cedar** are stored while they are being developed. One of the jobs of the release process is to move things from <PreCedar> to <Cedar>.

[Indigo]<PreISL>: The analogous development directory for **[Indigo]<ISL>**.

input focus: Suppose that the user types a key, while operating in an environment that supports multiprogramming lots of things going on at once, each in their own window. Which program was the keystroke intended for? Different systems have different conventions on this important point. In **Tajo**, the window in which the **cursor** is currently located gets the keystroke. But in several other systems, including **Cedar**, there is a concept called the ''input focus'' that is passed around among the running programs; whatever program has the input focus gets the keystrokes. Left clicking a mouse button inside of a window often has the side effect of giving that window the input focus.

Inscript: A mechanism for keeping track of user input to a program in a general way (key strokes, mouse clicks, and the like), used within **Cedar**.

install: A term applied to the Alto Operating System and a number of **subsystems** (notably **Bravo**), referring to a procedure whereby certain configuration options are established. Frequently, what is really going on is that the program being installed is salting away somewhere the current hard disk addresses of the pages of important files, so that later access to those files can avoid the tedious operations of looking up the file in a directory and chaining through disk headers to get to the right place within the file.

Intelnet: The Xerox corporate phone system, accessible by starting your dialing with the digit 8. Not to be confused with the **Internet**.

interface: A formal contract between pieces of a system describing a collection of services to be provided. A provider of these services is said to ''implement the interface''; a consumer of them is called a ''client of the interface''.

interface module: In **Mesa** and **Cedar**, interfaces are written down as a special kind of source file, starting with the word ''DEFINITIONS'' instead of ''PROGRAM''. This explicit encoding of an interface is called an interface module.

Interlisp: A dialect of Lisp with a large library of facilities, as witnessed by Interlisp's famous 15-pound reference manual (would that Cedar were so well documented!).

Interlisp-D: An implementation of Interlisp on **D-machines**, done by a group within PARC. It provides network facilities and high-level graphics primitives. See the **blue-and-white** report entitled ''Papers on Interlisp-D'', number CIS-5 (SSL-80-4) Revised.

Internet: Many Ethernets connected by Gateways form an Internet.

InterPress: A print file format standard that is currently under development: a second cut at the same issues addressed by **Press** format.

InterScript: A standard format for the interchange of editable documents that is currently under development.

Iris: An IFS server in **SDD**/Palo Alto, which serves as the official source of released **Pilot**s.

ISL Acronym for Imaging Sciences Laboratory, a part of **PARC** located on the second floor of **Building 35.**

Ivy: An IFS server in **PARC**, used by **CSL** and **ISL** mainly to store personal files.

jaggies: The annoying sharp corners visible when smooth curves are imaged on a raster device without sufficient resolution.

JaM: Acronym for John (Warnock) and Martin (Newell). An interactive language, similar to the language Forth, with a simple, stack-oriented execution model and equipped with lots for graphic operations as primitives; implemented in **Mesa**.

JaMGraphics: A component of an **ISL** release which provides **JaM** commands for all the **CedarGraphics** features. Creating JaM pictures with JaMGraphics can be very addictive.

Jedi: A **Hornet** at **PARC**.

Juniper: (archaic) An Alto-based distributed file system, built within **CSL**.

Juno: A constraint-based system for interactive graphics in **Cedar**.

junta: A technique for eliminating layers of the Alto Operating System that are not required by a particular subsystem.

Kanji: A **Dover** in **Building 34.**

Klamath: A forthcoming version of **Pilot** and other **Mesa** system software.

Lampson: A unit of speech rate. 1 Lampson is defined to be Butler's maximum sustained speed. For practical applications, the milliLampson is a more appropriate unit.

Larch: A family of specification languages.

Lark: The **Etherphone** processor, as well as the program that it runs.

Laurel: An Alto-based, display-oriented program that provides access to the facilities of **Grapevine** for sending and receiving mail. Succeeded by **Walnut** in the **Cedar** enviroment.

Leaf: A page-level file access protocol supported by some **IFS**'s.

level: There is a tree structure imposed upon the **node**s that make up a **Tioga** document, and the Tioga editor can be informed to suppress the display of all nodes deeper than a certain level. In combination with **scroll**ing, the levels commands in Tioga provide a convenient way to navigate in a well-structured document.

level i system: (for i ∈ [1..3]). A terminology for classifying (software) systems according to their intended user community:

1	implementers only
2	implementers and friendly users
3	naive users

Librarian: A **Tajo** program for check-in/check-out of the modules of a large **Mesa** system, used in **SDD**; also, a server for this program.

Lilac: A **Puffin** located in **CSL**, right next to **Clover**.

Lily: A program that provides teletype-style access to the mail sitting in one's **Grapevine** mailbox. Lily is designed to help out those folks who, because of travel or whatever, are unable to use their personal computers and either **Laurel**, **Hardy**, or **Walnut**. Also, a server that runs this program.

logical volume: A portion of a physical volume that is being used to support a **Pilot** environment: the **Pilot** equivalent of a **partition.**

look: An attribute of a character or string of characters in various editors, including **Bravo** and **Tioga**. ''Bold'' and ''italic'' are examples of Bravo's typographic looks, while ''emphasis'' and ''quotation'' are examples of the functional looks espoused by Tioga. The meaning of looks in Tioga, like the meaning of **format**s, is defined by the **style**.

Loops: A layer of software on top of **Interlisp** that turns it into an **object-oriented** environment tailored for building rule-based expert systems.

Lotus: Internal development name for the 1075 Xerox copier.

Lupine: The translator used to generate **RPC** stubs so that **Cedar** modules can call procedures located on remote machines.

Maggie: A tape server; that is, a machine on the **Internet** with tape drives that it will let a requesting machine use.

Magic: Acronym for Multiple Analyses of the Geometry of Integrated Circuits. A system for dealing with VLSI designs: printing them, converting them among formats, examining them with various programs.

Maintain: A program for updating **Grapevine** registration information. There are two versions of Maintain. One, with a widely reviled teletype-style user inteface, is available within **Laurel**, or as a **Tool** in **Tajo**. It is documented in the file [Ivy]<Laurel>Maintain.Press. The other, with a nifty buttons-style interface, is available in **Cedar**. It is not yet documented.

maintenance panel: An area on the chassis of a **Dolphin** or **Dandelion** that sometimes displays a three or four digit number. On **Dorado**s running **Pilot**, the maintenance panel number

appears in the cursor.

MakeConfig: A program that reads **Mesa configs** and **bcds** and produces a collection of commands that will compile and bind the many modules of a system in the correct manner to build a consistent system.

Marion: A **Librarian** server in **SDD**/Palo Alto.

Markup: A dead Alto **subsystem** for editing **Press** files.

MAXC: Acronym for Multi-Access Xerox Computer (pronounced ''Max''). A locally produced computer that is functionally similar to the DEC PDP-10. At one time, there were two **MAXC**'s, named Maxc1 and Maxc2, but Maxc1 has gone away forever. From now on, ''Maxc1'', ''Maxc2'', and ''Maxc'' are all names for the same machine, which used to be called Maxc2.

[Maxc]<Alto>: A directory on which standard Alto (**BCPL**) programs and subsystems are stored. Only object code files (extension **.br**) and runnable files (extension **.run**) are stored here; source files and documentation are stored on **[Maxc]<AltoSource>** and **[Maxc]<AltoDocs>**, respectively.

[Maxc]<AltoDocs>: A directory on which documentation for Alto programs is stored. Common extensions are **.press** (for files directly printable by **Press** or **Spruce**), and **.tty** (plain text).

[Maxc]<AltoSource>: A directory on which source versions of standard Alto programs are stored.

[Maxc]<Forms>: (archaic) A directory containing files that are usable as templates (in **Bravo**) for various kinds of documents (e.g., memos, letters, reports).

[Maxc]<Printing>: A directory containing Alto printing and graphics programs.

[Maxc]<PrintingDocs>: A directory containing documentation related to printing and graphics facilities such as **Press** files and font file formats.

[Maxc]<SubSys>: A directory containing standard **TENEX subsystems**.

MDG: Acronym for Methodology Discussion Group. A series of seminars or discussions on issues related to programming methodology that is held in CSL on Thursdays at 11 a.m.

Menlo: A **Dover** located in **ISL**.

menu: A collection of text strings, buttons, or icons on a display screen generally used to represent a set of possible actions.

Mesa: A PASCAL-like, strongly typed, system programming language developed by **CSL** and **SDD**.

Mesa Development Environment: The package of software used by **SDD** to develop other software in **Mesa**; combines the **Tajo** user interface with the compiler, **binder**, packager, and other system software running on top of **Pilot**. The name ''Mesa Development Environment'' is often used when the plans to market this body of software running on **Dandelion**s are being discussed.

MesaNetExec: A **Mesa** implementation of the **NetExec**; valuable because it knows how to load **Othello**.

MetaFont: A font-designing language built by Don Knuth at Stanford, and used to generate fonts for use with **TEX**. **Metafont** is available as MF.Sav on **Maxc.**

Microswitch keyboard: Microswitch is a company that makes keyboards. The standard **Alto** keyboard, also in use at **PARC** on **D-machines**, is made by Microswitch.

MIG: An acronym for Master Image Generator: a high-resolution laser-scanning printer, based on a photographic process. The MIG-1 can run up to 2000 bpi, while the slightly different MIG-3 runs at about 800 bpi. Also called the **Platemaker**.

Mockingbird: A music system that runs on a **Dorado** with an attached audio synthesizer and its keyboard. The goal of **Mockingbird** is to relieve the serious composer of some of the clerical burden of writing out scores for music as it being composed. For more details, see the

blue-and-white report ''Mockingbird: A Composer's Amanuensis'', number CSL-83-2.

mode: A special state through which certain user interfaces must pass in order to perform certain functions. For example, in order to insert characters into a document in **Bravo**, one must type the ''I'' key, which invokes the ''Insert'' command. The effect of this command is to put Bravo into ''insert mode'', in which typing the ''I'' key has a quite different effect (to whit, it inserts an ''I'' into the document). One must then hit another special key, ''ESC'', in order to leave ''insert mode''. Modes are locally viewed as generally evil.

modeless: Describes a user interface that is free of **mode**s. In such an interface, pressing a particular key always has essentially the same effect. **Laurel** was the first local system with an approximately modeless editor interface; the **Tioga** editing interface is very similar.

mouse: A type of pointing device with which many personal computers come equipped. The switches on the mouse are called ''buttons'' to distinguish them from the ''keys'' on the keyboard.

mouse-ahead: Analogous to typeahead, except refers to mouse clicks rather than to key strokes. Can become very confusing to non-**wizards**, as there is no analog of the backspace key for mouse clicks, that is, no way to cancel unwanted mouse clicks.

Multibus: An Intel standard specifying the physical and electrical characteristics of a bus by which various boards in small computers can communicate. Many useful boards that plug into a Multibus are available, such as Ethernet cards and disk controller cards. The **Dicentra** is a **D-machine** that uses the Multibus.

name lookup: In the context of network communications, the process of mapping a string of characters to a **network address**. Also, the protocol that defines the mechanism for performing such a mapping.

name lookup server: A computer that implements the **name lookup** protocol.

Nebula: A time server on the **Internet** that is equipped with an antenna to listen to time broadcasts made by a synchronous satellite, and hence has excellent long-term reliability. There is a display showing Nebula's opinion of the time in the same room as Clover: just the thing for setting your digital watch.

Neptune: An Alto **subsystem** providing more sophisticated manipulation of the file directory system than is available with the **Executive**. See also **DDS**.

NetExec.boot: A mini-**Executive** usable on an Alto without a spinning disk and obtainable directly over the **Ethernet** (from a **boot server**). The NetExec makes available a number of useful stand-alone programs, including **CopyDisk**, **Scavenger**, **FTP**, a number of diagnostics, and lots of neat games.

network address: A pair of numbers <network number, host number> that uniquely identifies any computer in an **Internet**.

nine-wire interface: A specification of how certain printers talk to their controllers.

node: A chunk of text in a **Tioga** document: each heading and paragraph in a document froms a node, and the nodes are hierarchically structured. Node-structured documents are easier to browse, using the **level**s commands in Tioga. Note: you can't have two nodes on the same line.

NS: An acronym for Network Systems: the protocols for using the **Ethernet** in the **Star** world. NS packets are analogous to **PUP**'s, and the NS protocols include analogs to such higher-level protocols as **FTP**.

Nucleus: A new virtual memory and file system base that is being built for **Cedar**, to replace portions of **Pilot**; it will emerge in Cedar 5.0.

Nursery: A large room in **CSL**, across from the Commons; so named because it was to be where new printers would be nursed to life, and also where fresh blood (summer interns and the like) would be housed. Does this mean that Bob Taylor thinks of graduate students as infants? I don't think

so; course, I could be wrong... The funny windows were intended to make it convenient to hold demonstrations in the **Nursery** with some of the audience on the outside, looking in.

object-oriented: Describes a philosophy about how programs should be structured that finds its purest expression in the **Smalltalk** system. An object is a little pile of private data together with a collection of procedures by which other folks are allowed to ask the object to do something. Other folks must not play with the data directly, but instead are required to interact with the object only by calling its procedures (or, in **Smalltalk** parlance, sending it messages.) Think about complex numbers as a trivial example: A non-object-oriented programmer would probably represent a complex number as a record containing two real numbers. An object-oriented programmer would be tempted to represent a complex number as a record containing public fields and private fields. The values of the public fields would be procedures, with field names such as: AddToMe, MyXCoord, MyYCoord, NegateMe, MyMagnitude, and the like. The private fields in the standard implementation of complex number would be simply two reals, named X and Y. The advantage of the object-oriented approach is that someone else can come along later and implement a new flavor of complex number that uses polar coordinates in the private fields, and previous programs that dealt with complex numbers will not have to be changed.

OIS: An acronym for Office Information Systems: a name for a concept, a type of product, and (perhaps) a market, not a particular organization.

OPD: An acronym for Office Products Division, located mostly in Dallas. They make and sell 820's and the like; see **products**.

Orbit: A high performance Alto-based image generator designed to merge source rasters into a raster output stream for a **SLOT** printer (e.g., **Dover**). So named because it ORs bits into buffers.

OS: Acronym for Operating System. Generally used to refer to the Alto Operating System, which is stored in the file **Sys.boot**. Rarely used locally to refer to the operating system of the same name that runs on IBM 360/370 computers.

OSD: An acronym for Office Systems Division, of which **SDD** is a part; they deal with the higher end of the office market, in contrast to **OPD**.

Othello: A network-bootable **Pilot** utility, good for initializing logical volumes and the like.

page (on a disk): A unit of length: an Alto or Pilot page is 512 bytes, while an **IFS** page is 2048 bytes.

PARC: Acronym for Palo Alto Research Center.

partition: A chunk of a large local disk that is being used to emulate the largest system disk that the **Alto OS** allows. A **Dorado** has five **partitions**, while a **Dolphin** has two. **Partitions** are numbered starting at 1; the phrase ''partition 0'' refers to the current default partition. The current **partition** in use is determined by the contents of some registers that belong to the disk microcode. You can change these registers with the ''partition.~'' command available in the Executive and in the NetExec. A (14-sector) **partition** has 22,736 Alto **page**s (11.6 MBtyes). It took a little adroit shoehorning to fit two full partitions onto a Dolphin's disk: it turns out that a Shugart 4000 has just one too few cylinders to squeeze in two full partitions. So we have to ask the heads to seek off the end of the advertised disk (on the inside, it happens), and put one more cylinder in there! Ah, the joys of hardware hacking...

PasMesa: A program that more or less compiles Pascal source into **Mesa** source, and hence assists in importing Pascal programs into our environment; developed in **CSL.**

path name: The complete name of a file, including the **file server** and **directory** or subdirectory on which it is stored everything you need to know to get the file. In the old style of writing (**Alto** and **IFS**), a **path name** consists of a machine name in square brackets followed by a directory name in angle brackets, optionally followed by one or more subdirectory names separated with right angle brackets, followed by the file name itself, as in

[Indigo]<Cedar>Documentation>BriefingBlurb.press.

Starting with **CIFS** in **Cedar**, a simple slash may be used instead of the various flavors of brackets, as in

/Indigo/Cedar/Documentation/BriefingBlurb.press.

PD files: A Printer Dependent print file format. The format and semantics of PD files are simpler than those of **Press** files. Software exists to turn **InterPress** masters into PD files, and also to print PD files on various marking engines, including **Lilac**, **Stinger**, and the **Platemaker**.

Peanut: A mail program in **Cedar** that fetches your messages into a structured **Tioga** document, rather than storing them in the **Cypress** database as does **Walnut**.

Penguin: Generic name for a type of 384 bpi laser-scan printer built on the Xerox 5400 xerographic engine, and connected to an Alto by means of an **Orbit** interface. **Penguin**s have better **solid-area development** than **Dover**s, and can also print two-sided. They are normally driven with **Spruce.**

Phylum: An **IFS** in PARC**.**

physical volume: The name for a disk pack in **Pilot**.

PIE: Acronym for Personal Information Environment. Implemented in Smalltalk, PIE uses a description language to support the interactive development of programs, and to support the office-related tasks of document preparation, electronic mail, and database management. For more information, browse [Ivy]<PIE>.

Pilot: An operating system that runs on **D-machines**, and was produced in **SDD** for use by **Star** and future products. Using Pilot instead of the Alto OS gives you the advantages of multiprocessing and virtual memory. Pilot is the current base for **Cedar**, although parts of Pilot will soon be replaced by the **Nucleus**.

pixel: A contraction of the phrase ‘‘picture element’’, referred to the tiny, usually square cells out of which a raster image is built up.

plaid screen: Occurs when certain kinds of memory smashes overwrite the display bitmap area or control blocks. The term "salt & pepper" refers to a different pattern of similar origin.

Platemaker: Another name for the **MIG.**

PolyCedar: A name for the polymorphic language in the Algolic tradition that is the subject of the religious material in the **CLRM**. A possible future project in **CSL** to design and implement such a language.

Poplar: An interactive programming language system implemented in **Mesa**, an experimental system in the direction of programming by relatively inexperienced users. Useful for text manipulation applications.

Poseidon: A **Tool** that provides the functionality of **Neptune** in the **Tajo** environment.

Press: A file format used to encode documents to be transmitted to a printer. Files in this format are conventionally given the file extension **.press**. Also, a printing server program, written in **BCPL**, that can print curves and raster images as well as characters and rules.

PressEdit: A subsystem that recombines **Press** files on a page-by-page basis; it can also merge illustrations into documents, although requesting this is a somewhat arcane and delicate operation.

primary selection: A chunk of text that has been distinguished, usually by mouse clicks, as an argument to a future editing operation. The current primary selection is indicated in **Tioga** by a solid underline, or by video reversal.

PrincOps: The Xerox **Mesa** Processor Principles of Operation, essentially a description of a particular **abstract machine**. **D-machines** implement the **PrincOps** architecture by means of hardware and microcode, and **Pilot** was constructed to run on **PrincOps** machines.

print server: A computer that provides printing services, usually for files formatted in a particular way. The term also refers to the software that converts such files into a representation that

can be processed by a specific printer hardware interface. **Spruce** and **Press** are examples of print server programs that accept the **.press** print file format.

proc: (or PROC:) An abbreviated form of the common and important word ''procedure''.

products: The following is a list of the most commonly encountered Xerox product numbers and their distinguishing characteristics:

600's	Memorywriters; from smart typewriters up through terminals/printers
820	personal computer product
860	display-based, word-processing terminal
1000's	new series of copiers being advertised with Marathon theme
1100	a Dolphin, sold outside to run Smalltalk and Interlisp
1108	a Dandelion, sold outside to run Interlisp
1132	a Dorado, sold outside to run Smalltalk and Interlisp
2600	desktop copier
3100	3 sec/page copier, good solid black-area development
4500	1 sec/page copier, 2-sided copying
5400	1 sec/page copier, good resolution
5700	1 sec/page laser-scan printer
6500	20 sec/page copier, color copying
7000	1 sec/page copier
8000's	the parts of **Star** have numbers in this range
8700	offset-quality, 1 sec/page, laser-scan printer
9200	offset-quality, .5 sec/page copier
9700	offset-quality, .5 sec/page, laser-scan printer

PSG: Acronym for Printing Systems Group, a part of Xerox.

public interface: An interface that offers to provide services to all comers. Private interfaces, in contrast, specify the services that various modules in a single program will supply to each other.

Puffin: Generic name for a type of 384 bpi laser-scan color printer built on the Xerox 6500 xerographic engine, and normally driven by **Press**.

PUP: Acronym for PARC Universal Packet. The structure used to transmit blocks of information (packets) on the **Ethernet**. Also, one such unit of information: a datagram. Bob Metcalfe once remarked that this name was chosen since all prior PARC communication protocols were ''real dogs''. See the **blue-and-white** report entitled ''Pup: An Internetwork Architecture'', number CSL-79-10.

Purple Lab: An air-conditioned machine room on the second floor of **Building 35**.

Quake: A **Dover** on the first floor of **Building 35.**

Quantum: Brand name of certain disk drives.

Quoth: A **Raven** in **ISL** (as in ''Quoth the raven . . .'').

R-name: A complete name from **Grapevine**'s point of view: **R-names** have two parts, a prefix and a registry, separated by a dot as in ''Anderson.PA''. **R-names** that designate distribution lists have prefixes that end in an up-arrow, as in ''CSL^.PA''.

Raven: A 300 bpi laser-scan printer on which the 8044 product **print server** is based, with good

solid-area development. Upgraded in **ISL** to 384 bpi and used as a **Press** printer.

registry: A concept used by **Grapevine** to partition the space of names. ''PA'' and ''WBST'' are examples of registries.

release: A consistent set of versions of all of the files in a large software system. **Cedar** releases occur whenever major enhancements in functionality become available or when sufficently numerous or important errors (see **show-stopper**) have been corrected.

release master: The person in charge of coordinating a **Cedar** release, with the help of special software (the ReleaseTool) based on **DF files**.

religious: Used locally to refer to a debate about which people have strong feelings, but for which there is no easy technical resolution; when discussing religious issues, positions are advanced based on belief rather than on understanding. For example, the question of whether or not **window**s in a user interface should be allowed to overlap and partially obscure each other, as pieces of paper do in the real world, is often the subject of religious debate. More experience in user interface design, or sufficient advances in the cognitive psychology of user interfaces, may someday make this question less religious.

Rem.cm: A file used by the Alto **Executive** to store commands to be interpreted after the current one has completed. See **Com.cm**.

replay: Refers to a Bravo facility that permits recovery after a crash. See **BravoBug**.

Reticle Generator: A version of the **MIG** that prints directly on masks for integrated circuits.

reverse engineering: Designing something by taking measurements from an existing sample that someone else designed.

Rigging: A **component** of **Cedar** that implements the various flavors of strings, including **Rope**s.

RockAndRoll: Another **Raven** printer in **ISL**.

Rockhopper: A **Penguin** in the **Bayhill** building.

RollBack: The way to return to a clean **Cedar** world saved by a **checkpoint**.

Rope: An immutable string of characters (a rope is a ''thick'' string). Ropes are the standard way to pass strings around within **Cedar**; other types of strings, including REF TEXT and REF READONLY TEXT, are available for places where performance is a big issue.

RPC: Acronym for Remote Procedure Call, a technique for calling a procedure from one machine to be executed in another machine over a network. Also, a package of software supporting Remote Procedure Calls within **Cedar**. RPC is the standard way for Cedar programs to communicate over the network: **Tank**, **Football**, **Alpine**, and **Etherphone**s all communicate by means of RPC. For more details about the concept of RPC, as well as fascinating references to life in the South Pacific, read Bruce Nelson's thesis, which is available as the **blue-and-white** number CSL-81-9.

Rubicon: The release of **Pilot** upon which **Cedar** is currently based.

rule: A printing term describing a rectangle whose sides are parallel to the coordinate axes; usually thin enough in one dimension or the other to be thought of as a (horizontal or vertical) line.

Scavenger.boot: An Alto program available through the **NetExec** that checks for damaged file structures in a **BFS** and tries to repair them.

SCG: Acronym for Software Concepts Group, a part of PARC. The builders of **Smalltalk**.

scroll: Refers to a method of repositioning text on a display as though as though one were moving a **window** over a long, continuous sheet of paper.

scroll bar: A **bar**, usually located along the left edge of a **window**, with the property that left or right **click**ing in this bar causes **scroll**ing to happen (middle clicking causes **thumbing**).

SDD: Acronym for System Development Division; the technical (as opposed to marketing) portion of **OSD**.

secondary selections: A chunk of text distinguished, usually by mouse clicks, as the second argument

to a future editing operation. The current secondary selection is indicated in **Tioga** by a gray underline, or by a gray background.

Semillon: A Grapevine server in **Building 35**.

server: A computer dedicated to performing some collection of service functions for the communal good (e.g., a **print server**).

seven-wire interface: Yes, Virginia, hardware people use the concept of **interface** as well as software folk. The seven-wire interface describes how the microprocessor located in the terminal of a **D-machine** (in the base of the CRT, to be specific) communicates with the parent computer.

show-stopper: A **bug** serious enough to prevent further progress.

Shugart: A subsidiary of Xerox that manufactures disk drives.

Sierra: A recent release of the **Mesa Development Environment,** based upon **Trinity Pilot**.

signal: A mechanism for handling exceptional conditions that arise in **Mesa** or **Cedar** programs. See **catch phrase**.

SIL: Acronym for Simple ILlustrator. An illustrator program used for logic design and drawing in general. A weird but efficient user interface; solid performance.

SIS: Acronym for Scientific Information Systems; the name of that part of **EOS** that is still a part of Xerox.

SLOT: Acronym for Scanning Laser Output Transducer.

Smalltalk: An integrated programming system based on **object-oriented** style and message passing, invented and developed by **SCG**. Described in great detail in a recently issued book(!).

SModel: A program that stores files back to remote file servers from one's local disk; SModel reads ''.df'' files in order to figure out what files have been changed, and which of these should be stored, and where in the great wide electronic world to store them. Use of SModel (confusing as it may be at the outset) is to be recommended over use of either **FTP** (in the **Alto world**) or the **FileTool** (in **Cedar**), since the version control and system-description features of ''.df'' files are very valuable.

solid-area development: The ability of a printer to produce large areas of black. Requests for large black areas on printers like **Dover**s, which don't have this ability, will result in a fringe of dark gray around a sea of light gray.

SophtSpheroid: A small, round, white object usually found on diamonds. Consider joining a Xerox softball team for more information on this indelicate topic.

Spruce: A program that takes **Press** files consisting of text and **rules**, converts them to a form acceptable by an **Orbit** interface, and prints them. A print server.

Spy: A program to investigate another program's performance when running in **Cedar**.

Squirrel: A personal database program based on the **Cypress** database in **Cedar**.

Star: An **OIS** product of Xerox, developed within **SDD.** Also referred to by various product numbers in the 8000's. The primary professional workstation of **Star** is the 8010, which uses a **Dandelion** processor.

Stinger: A **Hornet** located in **ISL**, running **Press**.

STP: The **Pilot interface** to the **FTP** file transfer protocol.

style: A collection of little programs in a language very like **JaM** that define the meanings of the various **look**s and **format**s of the text in a formatted **Tioga** document. Different style rules exist for how things should look on the screen and for how they should look when printed on paper (implemented by the **TSetter**).

subdirectory: File directories on an IFS can be divided into a hierarchical collection of subdirectories. The subdirectory names are listed from the root of the tree down to the leaves, and are separated by the single character ''>''(see path

subsystem: A program running under a specific operating system. Normally used to refer to Alto

programs that run under the Alto **OS**, but also used to refer to PDP-10 programs that run under **TENEX**.

Swat: A debugger used primarily for **BCPL** programs. Also, the key used in conjunction with the ''control'' or ''shift'' keys to invoke this debugger, as well as various other debuggers. The Swat key is the lowest of the three unmarked keys at the right edge of the keyboard. Used as a verb to refer to the act of striking these keys or entering the debugger.

Swatee: A file used by debugging programs (both **Swat** and the Alto/Mesa debugger) to hold the core image of the program being debugged. Also used as a scratch file by many Alto subsystems. Not to be deleted under any circumstances.

Sys.boot: An Alto disk file containing the executable representation of the Alto Operating System.

SysDir: The Alto file directory. Roughly speaking, this file contains the mapping from file names to starting disk locations.

SysFont.al: An Alto screen font used by the **Executive** and (generally) as a default by other programs. The safest way to change your SysFont is with the Delete.~ and Copy.~ commands of the Alto Executive. Simply FTP'ing a new font on top of SysFont will cause exotic behavior during the CounterJunta when FTP is finished.

system models: A part of the **Cedar** project, aiming at giving programmers help is describing the structure of large systems: getting consistent versions of files, replacing single modules within a running system, and recompiling and rebinding just what has been changed, all in the right order.

Tajo: The user interface portion of the **Mesa Development Environment** Each facility in the **Tajo** environment is called a **Tool**, and **Tajo** itself is sometimes called the **Tools Environment.**

Tank: An *n*-player video arcade game in **Cedar**. Get a tank game going and then close the tank **viewer** and check out the wonderful **icon** that results.

tasking: The technique by which the processor of an **Alto** or of a **D-machine** is shared between servicing various I/O devices and running the user's program (the **emulator**). Each I/O device is serviced by one or more tasks, which run at fixed priorities.

teledebug: Debugging one machine from another other the **Internet**. The prefix ''tele-'' is used in general for doing things remotely.

Telnet: A **PUP**-based protocol used to establish full-duplex, teletype-like communication with a remote computer. (The term is borrowed from a similar protocol used on the Arpa network.) **Chat** speaks this protocol.

Tenex: An operating system for the DEC PDP-10 computer, which also runs on **MAXC**.

TEX: A document compiler written by Don Knuth at Stanford; there are one and a half implementations of **TEX** at **PARC**: one in Sail that runs on **Maxc**, the half in **Cedar** (waiting on progress on the **Imager**). **TEX** can handle mathematical formulas, but doesn't let you see anything like what you get.

Thrush: The **Etherphone** control server; various **Lark**'s talk to Thrush in the process of setting up a call over the **Ethernet**.

thumbing: A technique of positioning a file (usually text) to an arbitrary position for viewing on a display. The name is intended to suggest the ''thumb-index'' with which some dictionaries are equipped, which performs somewhat the same function: gets you to roughly the right place quickly.

Thyme: An electrical-level dircuit simulator, used for evaluating the correctness and performance of small pieces of the designs of integrated circuits.

TIC: Acronym for Technical Information Center; the fancy name for what is more generally known as the PARC library.

Tioga: The document editor in **Cedar**, which was built by folk in **ISL**. Tioga formatting uses the concepts of **level**, **node**, **look**, **format**, and **style**; for more details, read TiogaDoc.tioga.

Documents formatted with Tioga can be printed with the **TSetter**.

TiogaDoc.tioga: Documentation for the Tioga editor. At one point, the official home of this file was the directory [Indigo]<Tioga>Documentation>.

TIP: A system for interpreting keyboard and mouse actions and turning them into sequences of commands. You may customize your Tioga user interface by layering your own TIP table on top of the standard Tioga TIP table.

Tool: A facility available in the **Tajo** environment, or the program that makes that facility available. For example, one speaks of the ''File Tool'', which can perform file transfers for you.

Tools Environment: Former name for **Tajo.**

transaction: A collection of reads and writes of shared data that is guaranteed to be atomic: either all of the writes happen (the **transaction** *commits*) or none of them do (it *aborts*). Furthermore, the reads will see consistent data in that either all of the writes made by some other **transaction** will be visible, or none of them will.

Trident: The brand name of a type of disk drive that is quite common around here. There are T-80's (that is, 80MByte Trident drives) and T-300's. Tridents are manufactured by Century Data Systems, a subsidiary of Xerox.

Trinity: The version of **Pilot** and other **Mesa** system software between **Rubicon** (the current base of **Cedar**) and **Klamath**.

TSetter: The typesetting program for **Tioga** documents; converts foo**.tioga** into foo**.press**, and optionally sends the latter to your favorite **print server**.

typeahead: An ability to type characters to a program before that program has asked for them. Useful for wizards; essential when using slow machines. See also **mouse-ahead**.

typescript: A file used to back-up information (usually text) appearing in a region of the display.

Twinkle: A **Gateway** in **Building 35** of **PARC**.

uncaught signal: An exceptional condition (perhaps an error indication) that no current program other than the **Mesa** or **Cedar** debugger has expressed a willingness to deal with. The debugger is willing to deal with anything, of course: it deals with these exceptional events by halting the offending process and then informing the user. In the language of the **CLRM**, an uncaught signal should be thought of as an invocation of a dynamically bound procedure that turns out not to have been bound at all; see **catch phrase**.

user: A person (rather than a program) who avails herself of the services of some program or system. At the moment, the author is a **user** of **Tioga**. See **client.**

user.cm: A file in the **Alto world** containing a number of logically distinct sections that each define certain configuration parameters (e.g., the location of a preferred **print server** for a particular file format). Programs that interpret such parameters are often organized to read user.cm only at **install**ation time (e.g., **Bravo**).

user.profile: A file in **Cedar** that specifies the default values of a collection of configuration parameters. Each user may modify these values however she wishes. The resulting personal profile information is stored in a file that is named ''<user's name>.profile''.

UserExec: The command interpreter for **Cedar**.

Versatec: A subsidiary of Xerox that makes electrostatic printers, as well as the ''Expert 1000'' computer-aided design system, which runs on a **Dandelion**.

viewer: The name for a window in the **Viewers** window package.

ViewerDoc.tioga: Documentation for the **Viewers** window package. You might try looking for this file on the directory [Indigo]<Cedar>Documentation>.

Viewers: A screen management and window package for **Cedar** providing **button**s, **menu**s, and **window**s.

ViewRec: A software package in **Cedar** that produces convenient user interfaces to fairly arbitrary

programs automatically.

Viking: A **Dover** on the first floor of **Building 35.**

VLSI: Acronym for Very Large Scale Integration of electronic circuits on chips.

VM: Acronym for Virtual Memory.

Voice: A small but mighty project in CSL to tame the telephone and otherwise make full use of voice communications in our personal information systems. The Voice Project recently produced the **Etherphone**.

Walnut: A mail system for **Cedar**. Walnut uses the **Cypress** database to store and organize messages, and it calls upon **Grapevine** to transport them.

Watch: A **Cedar** performance monitoring tool displaying computing activity.

WaterLily: A **Mesa** program that does source compares: compares two text files and reports the differences. Available in Alto/**Mesa**, **Tajo**, and **Cedar**.

wedged: Describes the state of a program when there is no response to input from either the keyboard or the mouse. May affect the whole system (*my system is wedged*) or just some part thereof.

weekly meeting: The (unimaginative) name of ISL's weekly meeting, held on Tuesdays starting at 11:00 am. See also **Dealer**.

whiteboards: A package in **Cedar** for arranging and accessing information graphically.

Winchester: Originally, this was the name of a project within IBM. But the name leaked out, and it is now used industry-wide to refer to a particular rigid disk technology. In a Winchester disk drive, the heads and platters come all hermetically sealed; that is, Winchester drives do not use removable disk packs.

window: A display region, usually rectangular, used to view (a portion of) an image that generally exceeds the bounds of the region.

wizard: One who knows a programming system inside and out.

Wonder: A **Dover** on the third floor of **Building 35.**

world-swap: The process of writing out the complete state of a machine's processer and memory onto a disk file, and of swapping in a different state. Some debuggers work by means of **world-swaps**, which swap between the debugger and the program being debugged. Note that, the more memory you have, the slower a world-swap will be.

XGP: (archaic) Acronym for Xerox Graphics Printer. An obsolete, CRT scanned, 200 bpi, continuous paper, xerographic printer.

XM: Acronym for Extended Memory: an option on Alto II's that allows the memory size to be increased from one to four **banks**.

Yoda: A **Dover** in **Building 35**.

Zinfandel: An Alto mail server that is part of the **Grapevine** distributed transport mechanism.