Abstract: This document is a general introduction to the Interlisp-D computing environment, slanted towards the needs and interests of newcomers to the Intelligent Systems Laboratory and Xerox Artificial Intelligence Systems. This is definitely in *DRAFT* form, since it still talks a lot about Cedar and hardly any about Interlisp-D!!!!

For Internal Use Only

Raison d'Etre

The purpose of this document is to help immigrants adapt to the local computing community. "The local community" primarily means Interlisp-D users within PARC's Intelligent Systems Laboratory and the development group of Xerox Artificial Intelligence Systems. Immigrants to other computing communities within Xerox may also find this document of interest no guarantees are made. I shall assume herein that said immigrants know quite a bit about computers 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. This document was "ripped off" from a similar one written for the Computer Sciences Laboratory of PARC, whose members primarily use another environment -- Cedar, in the great PARC tradition of developing many different programming environments.

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 <reference>, where n is a citation reference in the bibliography at the end of this document. Standard citations to the open literature appear as [reference].

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). 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, 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 Harvard Concise Dictionary of Music, or the Sunset Western Garden Book [sunset]; Grapevine servers are named after wines; Dorados are named after hotels, spices, philosphers or ships (depending on who owns them); XDE releases are named after California rivers. This convention about names does not meet with universal approval.

Local Hardware

Most of the offices and some of the alcoves around 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 "old" University Grant program (U. of Rochester, CMU, MIT, and Stanford), and other places. (There's a "new" University Grant program in progress.) 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 displas 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 [grayscale]. 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. Some mice have two buttons, others three. The three mouse buttons go by various names; "left", "middle", and "right" is one set of names. The mouse buttons have at some point also been 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 the blue-and-white report on the subject [opticalmouse].

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 [ethernet]. 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 <netmap> 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 t"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) [PUP]. The new ones are known by the acronym NS (Network Systems) [NS]. 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 [ALTO]. 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. It was sold by Xerox AI systems as the Xerox 1100 with Interlisp-D and Smalltalk.

The Dandelion

The Dandelion is the D-machine processor that is used in the Star products and the Xerox 1108. It comes in a box about half the width of an Alto chassis, and roughly the same height and depth. Dandelions are both faster and much cheaper than Dolphins. Dandelions talk only to 10 MBit Ethernets. Recently a number of hardware enhancements have been developed for Dandelions, including the ability to add a lot more memory (up to 4MByte) and hardware floating point, IBM PC Bus controller, etc.

The Dorado

The Dorado was (and probably still is) the most powerful personal computer around. The Computer Science Lab built the Dorado as 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 [DORADO]. Dorados come equipped with a 315 MByte Winchester drive; older models have an 80 MByte removable-pack disk drive at present. 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. Dorados are generally 3-5 times faster than DLions running Interlisp, except for some notable exceptions (Dorados have an incredible memory bandwidth, and so can move the bits around the screen *really* fast; the DLions have special hardware for floating point which makes them faster than the (microcoded) Dorados.). One primary difficulty about Dorados is that they're a lot more expensive than DLions, and don't fit in your office. As a result, while some people have personal Dorados, there are also *pool* machines. 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. (This case was known as the APC, or Armored Personnel Carrier, which it resembled.). 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 the *next* high-performance contender will come out again and live in your office.

The Dicentra

The Dicentra is another D-machine of which there aren't a lot around. 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 called the Wildflower.

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.

Other D-machines

There may well be other D machines brewing in laboratories and development groups. This document won't talk about them, as they are more closely tied with some product plans that should not be widely circulated.

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 <DORADOBOOT>. 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 <DORADOBOOT>.

DLions have a boot button located right under the Maintentance Panel. It's labeled B-reset. There's another button next to it, labelled Alt-B, which is used in conjunction with B-reset to boot the machine in various ways -- you hold down both, let up on B-reset, and the maintenance panel lights cycle with numbers 0000, 0001, 0002, 0003, etc. When you let up, you get the "boot option" that you let up on: 0000 (or 0-boot) usually boots Lisp from the disk (or just B-reset), 1-boot boots Mesa from the disk, 2-boot boots from floppy, 3-boot boots from ethernet, 4-boot boots something that I forget, 5-boot gets "diagnostics" from a floppy, 6-boot gets an alternate Ethernet Mesa, and a few others. (0010-boot with a floppy-cleaning-diskette in the floppy drive will clean the floppy heads. This is useful you have a broken machine with a broken ethernet, and you want to floppy boot diagnostics but can't because of dirty heads!)

There is one more bit of folklore about booting Dorados that's fun to mention--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 a 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.

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.

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.

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" or "XDE" or "Basic Workstation". This body of software may soon be marketed under the name "the Xerox Development Envirnoment". In addition, Pilot Mesa is the current base of the Cedar project in CSL. More on Cedar later. Although Mesa programs look a lot like PASCAL programs when viewed in the small, Mesa although Mesa provides and enforces a modularization concept that allows large programs to be built up out of smaller pieces. The Mesa language is described by a manual [MESA].

Smalltalk

Smalltalk was developed by the folk who now call themselves the Software Concepts Laboratory (formerly known as the Learning Research Group and then the Software Concepts 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.

[MORE]

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 [EPE]. After pondering the alternatives for a while, CSL chose to build yet another programming environment, based on the Mesa language. That new environment was named "Cedar". The programming language underlying Cedar is essentially Mesa with garbage collection added. Now, adding garbage collection actually changed 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 Interlisp world.

In Interlisp, the current best sources are the Interlisp Manual mentioned above.

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.

Some people believed that, because local file systems are small, it wasn't in general practical to store more than one version of a file on the local disk. Thus, some local file systems don't support versions, and in programs that use them,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 Interlisp programmers) gets burned by it at least once. Some text editors in XDE and the Alto world *do* 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.bravo" is stored in the file "foo.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

The phrases "living cleanly" 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 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.

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 nineteen. 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 -- it necessarily "boots"the machine.

When operating in the Pilot world, a disk pack is called a physical volume, and it is divided into worlds called logical volumes.

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 "*"s in 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!).

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, Tedit 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.

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 <PRESS> 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, called Interpress. The print servers that are part of the Star product speak a dialect of Interpress as will most other new Xerox printers. A print file in Interpress format is called a master. CSL's 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.

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.

CSL 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 <PDPRINT, PRESS>. 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 CSL's color Press printer and may be found in room 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 room 2301, there are 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 Interlisp, most folk use Lafite, whose documentation appears as <LAFITE.PRESS>. 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.

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.

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".

"Pop!"

THIS phrase 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

Most groups 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 people 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 machines 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

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.

[SUNSET]: 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.

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

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

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

[ETHERNET]: 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.

<TOPOLOGY>: [indigo]<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.

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

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

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

[ALTO]: 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.

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

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

[MESA]: Mesa programmers manual.

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

[22]: Interlisp Reference Manual. 1983.

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

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

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

[ALTO]: 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

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

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

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

<PRESS>: [Indigo]<PrintingDocs>PressOps.press. Describes the Press printing program.

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

[GRAPEVINE]: 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.

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

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