The Alto-Dolphin-Dorado Briefing Blurb

or

Exploring the Ethernet with Mouse and Keyboard

BY LYLE RAMSHAW

A revision of: A Field Guide to Alto-Land, by ROY LEVIN

MAY 1981

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

Raison d'Etre

Are you a programmer? Are you sick of manuals that tell you how to use a software system without telling you why it behaves as it does? Are you frustrated because you don't know the unstated assumptions behind the interesting discussions you hear around you? Have you ever wanted to browse through the source code or the documentation for a program, but couldn't figure out where to find it? If the answer to some of these questions is "yes", read on! These and other useful (and occasionally entertaining) tidbits shall be made known unto you.

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 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 program listings. 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*].

Reading a document from front to back can be mighty boring. This document is organized so that you can (supposedly) browse through and read the parts that look interesting. In fact, the current version of this document is so disorganized that it is not at all clear that there really *is* a front and a back in the normal sense! This means that the usual bottom-up approach to documentation (define your terms before you use them) has been abandoned. Instead, all the relevant terms, acronyms, and the like have been collected in a glossary at the end. Some information is contained *only* in the glossary, so you may want to scan through it later (or now, for that matter). It is 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.

Alto-Dolphin-Dorado-Land

Behind that inviting screen there lurks a wealth of fascinating history, folklore, and (occasionally) documented wisdom. However, even the great storytellers of old occasionally forgot that their attentive audiences included travelers from other lands who were ignorant of the local customs and traditions. So it was with the Alto gurus. What follows is a transcription of the oral history of the Alto culture acquired by a relatively recent settler in these parts. It makes no claim to completeness, balance, or fairness. (A separate document exhibiting these qualities may be found on {20}.)

Before exploring Alto-land, you should know something about the names of the creatures you will find there. The prevailing philosophy about naming systems in CSL and SDD is perhaps somewhat different from the trend elsewhere. While we have our share of alphabet soup (e.g., PARC, FTP, MAXC, IFS), we are trying to avoid making it any worse. Names for hardware and software systems are frequently taken from the *Sunset Western Garden Book* [14]; Grapevine servers are named after wines; Dorados are named after capital ships; Pilot releases are named after California rivers. Personal machines also have names that frequently, but not necessarily, come from the same kinds of sources. These names are chosen by the machine's owner, and registered with an entity named NetSupport.WBST. 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. Accordingly, we abandon the topic here and move on to more concrete matters.

Alto, Dolphin, and Dorado 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 1000 Altos in existence now, spread throughout Xerox, the four universities in the University Grant program (through which Xerox has given Altos to U. of Rochester, and Altos, Dovers, and IFS's to CMU, MIT, and Stanford), and other places. In contrast, the Dolphin and Dorado are newcomers to the scene. A Dolphin is somewhat more powerful than an Alto, and a Dorado is gloriously more powerful. Private ownership of Dorados is a recent innovation, and has not yet spread very far. Most of us get our fixes of Dorado time by signing up on lists to share one of the "pool" Dorados.

The Alto hardware

The genus "Alto" comprises two species, imaginatively named Alto I and Alto II, and there are several sub-species as well. All members of the genus have at least a display, a keyboard, a mouse with 3 buttons, and a processor/disk cabinet. The front cover of an Alto I cabinet has a ventilation grill just at the bottom, while on the front cover of an Alto II, the grill goes all the way up to the disk slot. All Alto I's have a main memory consisting of 64K 16-bit words, called one *bank* in local parlance. Most Alto II's are equipped with the extended memory option (XM), and have up to four banks of memory (that is, 256K 16-bit words). Needless to say, you can do a lot more with four times the memory.

The first Alto II's made came equipped with a funny keyboard, properly called the *ADL keyboard*, although sometimes also simply referred to as the *Alto II keyboard*. An ADL keyboard is larger, and has a different key layout, including columns of function keys on the left and right. The ADL keyboards were generally reviled, and hence did not stay in production long. Nowadays, they are pretty rare. If you want to see one, check out Clover's Alto. Don't be confused by documentation that warns you that keys will be in different places on an Alto II; you can safely ignore these

warnings unless, God forbid, your Alto II has an ADL keyboard.

The innards of the Alto are revealed in gory detail in a very complete manual {1}. Facts, figures, specifications, and programming information (at the machine level) are all there. What isn't there is a bit of the philosophy underlying the machine design and organization. In particular...

1) There isn't much special-purpose hardware in the Alto. Most of the nifty stuff you can read about in the hardware manual is in fact implemented by microcode. This gives us considerable flexibility in the way we design software interfaces for experimental devices and specialized instruction sets. In fact, Mesa and Smalltalk are implemented almost entirely with "special" microcode.

2) The display is rather different from a number of other common displays. Instead of containing a character or vector generator, the display hardware interprets individual bits. One bit in memory shows up as one dot on the screen. Since the screen is 606 by 808 points, a quick calculation shows that a full-screen display requires nearly half of an Alto I's memory. For a machine with only 64K of memory, that seems a big price to pay. The theory is that in exchange for the space we get enormous freedom to experiment with various strange ways of manipulating the screen.

So much for philosophy—how does it work out in practice? Well, excessive flexibility breeds chaos, so a number of things have been standardized. All Altos contain a ROM that defines the "normal emulator" (i.e., the standard instruction set) and the standard i/o device interfaces (e.g., display, disk, ethernet, and so-called "junk i/o"—the keyboard and mouse). The instruction set is derived from the Data General Nova, though the i/o structure is rather different and several specialized instructions exist to support various display manipulations. If you have a spare hour or so, read about BITBLT in {1}. Then try to imagine writing the microcode to implement it. Since the microcode for these standard facilities is blown into a ROM, suggestions for improvements/extensions are treated with considerable skepticism. Microcode hackers will find the additional 1K control RAM available on all Altos a reasonably comfortable sandbox in which to play. If you are both a microcode hacker and a concrete pourer, you can also use a second 1K of ROM on Alto II's. A few Alto II's substitute 3K of RAM for the second 1K of ROM, which gives you even more room to play without any need for concrete..

The display has enormous potential, and there are a number of programs around that exploit it in interesting ways. We also feel compelled to note that at least an equal number of programs still treat the display as a glass teletype. Home-made cookies require more effort, but they taste a lot better than the store-bought variety. Fortunately, more and more people are getting into home cooking.

The mouse has two obvious properties—it rolls and it clicks. Inside the Alto hardware, 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 are named red, yellow, and blue, even though physically they are nearly always black. This choice was made because not all mice have their buttons arranged in the same way. On some (older) mice, the buttons are thin, horizontal bars; the top one is red, the bottom one is blue. On most mice, however, the buttons are wide, vertical bars, with red at the left and blue at the right. Some people insist on naming the buttons red, yellow, and green—perhaps as kids they had strange paintboxes, or were fixated on traffic lights.

A somewhat unusual property of the Alto is that the keyboard and mouse buttons are unencoded; that is, there is a bit for each key that indicates whether the key is up or down. Many programs

distinguish between holding a mouse button down and clicking it down and up. Fewer programs play such tricks with the keyboard, although combinations of keys that would jam a conventional typewriter are quite meaningful to some programs, e.g., the NetExec. Fortunately, there is standard software that enables you to treat the keyboard in the usual way if you want to. Note that this "usual way" involves a feature called *infinite rollover* that changes your typing style after a while: you can have arbitrarily many keys depressed at the same time without causing a jam, and without causing any strokes to be discarded. Every key that goes down counts as a key stroke when it goes down, and when that key comes up again doesn't matter at all. A few weeks of typing on this sort of keyboard, and you begin typing common letter clusters such as "ion" with just a flick of the wrist; after all, if the letters come out in the wrong order on the display screen, you can always edit them later.

With a personal computer, you are programmer, system hacker, and console operator all rolled into one. If you don't like the state your program has reached, you can always press the boot button and start over—an option you rarely have on larger, shared machines. It is a good idea NOT to press the boot button when disk activity is in progress, however, since you may interrupt the writing of a disk sector halfway through, thus rendering that sector unreadable. However, the Alto differs from many small computers in that it lacks those time-honored, nitty-gritty debugging facilities: the console lights and switches. If things are so screwed up inside that you can't get some sort of (software) debugger running, there isn't much you can do as console operator. This tends to down-play the operator role and emphasizes the system hacker role. ("Let's see, if I hit shift-Swat, that will write the core image on Swatee, and if I then bootload the debugger...") It also makes certain kinds of bugs, e.g., those that smash crucial memory locations in low-core, very difficult to find.

If you have just turned on an Alto and are about to spin up your disk and do some work, it is a good idea to hit the boot button *before* your disk comes up to speed: When an Alto is first powered up, it comes up in an unknown state, and there is some small but nonzero chance that the processor is sitting in a loop just waiting to eat your disk (that is, write something on it).

Double Disks on Altos

Many Altos in CSL have been equipped with a second disk drive, which either balances on top of the processor cabinet or sits on some flat surface nearby. The two drives of a double disk Alto can be used in two different ways: either the two disks constitute separate file systems, or they together constitute a single, larger file system. It is possible to extend a one disk Alto file system into a two disk system at any time, but the reverse is not possible. And a two disk system will only operate when both of its disks are loaded into the drives of a double disk Alto. A double disk machine has a second little switch hidden at the back of the keyboard, in addition to the boot button. This switch serves to interchange the roles of the two drives, called DP0 and DP1. If you flip this switch while disk activity is in progress, you will fairly likely get what you deserve: a smashed page on one or both disks. The Scavenger may help you recover.

The Dolphin (formerly called the *DO*)

A Dolphin looks a lot like an Alto II. The biggest difference is that the Dolphin has a nasty looking piece of solid sheet metal where an Alto has a slot into which you can stick your disk pack. You see, a Dolphin comes with a Shugart 4000 disk instead of the Diablo 31 disk of the Altos—and the recording medium of a Shugart 4000 is not removable (well, not *easily* removable anyway: with the right tools and expertise, you might swap platters in an hour or so). This makes it somewhat harder to borrow someone else's Dolphin than it is to borrow their Alto—see the discussion below on *Living Cleanly.*

Dolphins have various advantages over Alto II's: Mesa programs run significantly faster; long pointers in Mesa work (making it easier to use lots of memory); the Shugart disk is bigger (two

partitions); and Pilot runs on Dolphins. Most Dolphins have wide displays, 1024 bits in width instead of the 606 bits of the standard Alto-style display. The extra screen space is only available to programs running in the Pilot world.

The Dorado (very formerly called the *D1*)

A Dorado doesn't look anything like an Alto, although most Dorados today come equipped with an Alto-style terminal (display, keyboard, and mouse). In fact, in a return to the ways of the past, a Dorado's processor is located in a remote, heavily air-conditioned machine room. When the Dorado was being designed, it was intended and hoped 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. These cases are fondly known as APC's, although no one is sure what the letters stand for in the Dorado's case (see the Glossary). But experience indicated that Dorados run too hot when inside of these cabinets, and the concept of having Dorado processors in offices has been abandoned. With VLSI and all, there is some hope that the Dorado's successor (the Dragon?) will once again come out of the machine room and into your office.

Dorados have varying amounts of memory, but at least 512K 16-bit words, or equivalently, 8 banks. In addition to these oceans of real memory, the Dorado has a fairly whippy processor, a cache, and a Trident 80 MByte disk; this all adds up to pretty impressive performance. A Dorado is roughly three to five times faster than an Alto when emulating an Alto, that is, running BCPL. And 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—yet (and the related fact that they are rather tricky to build).

Dorado disks can be changed with somewhat greater ease than Dolphin disks, but the process is still painful enough that changing disks is not the normal mode of operation. The biggest difficulty is that you must be at the processor to change the disk, 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 also need to borrow at least some of the space on that Dorado's local disk: this brings us to the issue of *Living Cleanly.*

Living Cleanly: local disks, file servers, and the like

Local disks are very convenient, but not very reliable. It is quite tempting to work along on an Alto for weeks at a time, without backing up your files on any other medium. You might be able to do this forever without getting burned. On the other hand, those of us who have been hanging around here for a while could tell you many sad stories of head crashes and dropped disks, that left our colleagues (or even ourselves) with disk packs suitable only for wall decorations. In fact, most of those very disks are now serving as wall decorations. Hence, the principle of clean living: you are living cleanly if your pulse and blood pressure would remain substantially constant when you are informed that your local disk has been degaussed. Make sure that all of the bits you really wouldn't want to lose are out some file server, such as Maxc or an IFS, where suitable precautions are constantly being taken by wizards to protect against disk failure. Appropriate use of command file, dump files, and other automatic aids can make this task easier.

Encouraging clean living has another benefit: shared local disks are only practical if everyone lives cleanly. The management of the public partitions on Dolphins and Dorados presumes a policy of clean living: when you are done working on a public partition, you must store away all of your files on remote file servers. It is polite to delete your files from the local disk as well, to give whoever follows you

more space to play; but this is not nearly as critical. If you find that there isn't enough space on a partition, you are perfectly within your rights to delete the random files that are lying around on the partition to regain space. And the creators of those files won't mind, it says here, because they have been living cleanly. In fact, there is an authoritative image of what a "completely clean" public Alto-Mesa 6 partition for a Dorado is out on [Ivy]<Cedar>Mesa6.bfs. If you find that the public Mesa partition you just started using seems to be screwed up in some way, you can always get back to a known good state with lots of free space by smashing the whole partition with that image, using CopyDisk.

(The above paragraph is the "letter of the law" regarding the sharing of public partitions. 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. This topic is discussed at greater length below...)

It would be nice if our computers could handle more of this file shuffling for us. There is some hope that we will be able to move someday to a glorious future in which all of the independent file system fiefs will be combined into a universal file system, and all local disks will be employed simply as automatically-managed caches for that file system.

A few comments on Booting

An Alto has one boot button, hidden behind the keyboard. On Dolphins, there are two boot buttons, one at the back of the keyboard, and one on the processor cabinet itself: the two perform roughly the same function, but the one on the cabinet is a little more potent. On Dorados, the situation is much more complex. There are really two computers involved, the main processor and a separate microprocessor called the *baseboard* computer. 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, and 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: {24}. 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 {24}.

Small Fish in a Big Pond—the Network

Two's company, three's a network. You can do a lot with an Alto, but at best it's still a classy minicomputer. With hundreds of them out there we should be able in theory to do all sorts of wonderful things. In practice, we actually do some of them. You should read the paper by Metcalfe and Boggs describing the *Ethernet* [2] for a good introduction to the communication network that connects our Altos together. In essence, a collection of Altos within reasonable proximity is hooked together by an Ethernet. Ethernets are connected to each other by Gateways, which for most purposes allow us to ignore the topology of the resulting network; the network as a whole is called the *Internet.* However, occasionally it's nice to know where things *really* are, and that's when a map {3} is helpful. For programs (which are notoriously poor map readers), the Gateways also provide an information service {15}.

We all know how uncommunicative computers can be when left to their own devices, and Altos are no different. That's why we invent careful protocols for them to use in talking to each other. Most of the protocols now in use on the Ethernet are called PUP-based (PARC Universal Packet) {16}.

Built on top of the Pup protocol are quite a number of others, some of which you can read about in {4,5,6,7}. You will probably hear some of the following protocol names being tossed about in conversation:

1) EFTP - stands variously for Early FTP, Ears FTP, Experimental FTP, Ether FTP, Easy FTP. A venerable protocol now mostly used to transfer files to printing servers. The Alto program EmPress uses it for this purpose.

2) FTP - refers to the File Transfer Protocol, as well as the Alto program that implements it and provides an interactive user interface. If you come from the Arpanet world, don't confuse this FTP with the one out there—ours is Pup-based and incompatible. On MAXC, where both the Pup and Arpa FTP protocols come in handy, the name FTP refers to the Arpa one and PupFTP (obviously) refers to the Ethernet one.

3) BSP - the Byte Stream Protocol. Built on top of Pup, this protocol is used by conversants who want to view the network as a full-duplex stream of 8-bit bytes. BSP is used to implement FTP and Chat.

4) The Grapevine Protocols. Grapevine is a distributed message transport system, and clients of this system, such as Laurel, use a special set of protocols to interact with it.

There are quite a few other protocols in use as well. The wisdom here is: if you have some nifty multi-machine communication to do, look around—someone may well have done your protocol work for you. There also are a number of communications wizards about who can keep you from falling into various traps.

Question: Why have a network? Answer: Because it's nice to be able to pawn off some of the dog work on other machines, leaving your Alto free to do the interesting stuff. That's why we have a number of machines generically called *servers*. Normally, servers have special purpose, expensive hardware attached to them (e.g., large-capacity disks, printers), and their sole purpose in life is to make that hardware available to more than one person/Alto. We tend to identify servers by function, so we talk about printing 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 Altos could communicate effectively with server Altos.

Printer servers and file servers get the biggest workout. There is considerable history and folklore surrounding printing and printing servers—you will find some of it in later sections of this document. One doesn't tend to hear much about file servers, except when they are down; nevertheless, they are essential to our computational well-being. Because Alto disks are rather small (another topic we'll come back to later), we rely heavily on file servers to store libraries of Alto packages, subsystems, and documentation. In Palo Alto, our primary file servers are Maxc and two IFS servers, Ivy and Iris. A file server also acts as the natural common repository for the "truth version" of a system being constructed by a number of individuals. And file servers are important to single individuals since they provide greater capacity and higher reliability than local disks.

Alto software

The first high-level programming language used on the Alto was BCPL, and quite a bit of program writing was done in that environment. But BCPL is now a dead environment. It will be around for some time to come, since there are BCPL programs that perform valuable functions for which

there are no current replacements: the printing server programs Press and Spruce are two important examples. Other languages that run on our computers are: Smalltalk [18] (an integrated, interactive, object-oriented programming system), Mesa [19] (a strongly-typed, PASCAL-like implementation language), Poplar (a simple, interactive, text-oriented language), a dialect of Interlisp, and others. Mesa is perhaps the most widely used, with Smalltalk and Lisp coming in second and third. Mesa was given a big boost in CSL when a variant of Mesa (itself called Cedar) was chosen as the base language for the Cedar programming environment.

There is a reasonable amount of introductory documentation for systems you commonly need (e.g., Bravo, FTP, the Alto Executive) in [13]. This is by far the most useful single reference for the Alto environment. Since the entire document can't be reprinted when any subsystem changes, however, you can expect the information there to be somewhat out of date at any one time. If you suspect you need the latest documentation for some program X, you might try [Maxc]<AltoDocs>X.press (or X.tty, or X-news.press).

The Alto Operating System

BCPL programs typically run on top of the Alto Operating System, which is itself written in BCPL. Like most OSs, this one provides a number of basic facilities, not all of which are needed by any one program. Because the small memory of the Alto is precious, a technique called "Junta" exists which permits BCPL programs to get rid of unneeded portions of the OS during their execution. "Counter-Junta" brings them back. You can read about the layers of the OS in {8}.

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 (e.g., the file system) have been (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.

There is a program called the Executive which runs on top of the OS and provides a command interpreter with a number of natural facilities, such as "tell me what files are on this disk", "run this program", "execute this command file", "go away". System maintainers will tell you that the Executive is "just another program—if you don't like it, you can write one yourself". That's true—you could also write a Mesa compiler yourself, but ... . In all fairness, however, the Executive is one of a number of programs that have reached the state where maintenance consists of aggravating details. Consequently, requests for feature enhancement are not likely to fall upon receptive ears. Perhaps the most useful features of the Executive are file name completion (ESC) and *-expansion, particularly in conjunction with subsystem invocation. You should also read about control-X and control-U in the Executive section of {10}. Also, check out key*i*.cm for $i=1$, 2, and 3.

No matter what program you are running, there are times when you want to say "get me out of this mess and back to somewhere more comfortable!" Unless things are *really* messed up, you have two choices, both of which require you to know where the Swat key is. (This invaluable key is unlabelled! On most keyboards, it is at the extreme lower right, next to the shift key; on ADL keyboards (in case you ever find one), it is at the extreme *upper* right, separated somewhat from the main keyboard area.) By hitting shift-Swat, you can normally get back to the Executive—note that you must use the *left-hand* shift key to type "shift-swat", not the right hand one: the right hand one is too close to Swat itself for comfort. Control-shift-Swat will normally get you into Swat (the BCPL debugger). What you do there is your own business (see {10}). Control-Swat will get you to the

Mesa debugger if you are in a Mesa program, and there is a Mesa debugger on your disk, and it is installed, and your core image isn't too badly screwed up. If you are just trying to abort whatever program you are running, you probably don't want to be in the debugger anyway. If these last-ditch facilities don't seem to work, things are very confused, and you will have to boot your Alto, using the little white button that is not only unlabelled, but hidden behind the keyboard.

Pilot

The folks in SDD have produced a new operating system, written in Mesa, called Pilot. Pilot does not run on Altos, but it does run on D-machines such as Dolphins and Dorados. Furthermore, by running Pilot on your D-machine you get all the advantages of virtual memory and multiprocessing, which you don't get while pretending that you're running on an Alto. Note that there are currently two different systems that someone can be talking about when they say "Mesa": Alto Mesa and Pilot Mesa.

The Network Executive

There are several facilities available over the Ethernet that do not require a disk. You can boot any one of these programs into your Alto by pressing down a strange collection of keys simultaneously and hitting the boot button, but there is an easier way. If you hold down the BS and ' (single quote) keys and hit the boot button, you get the NetExec, a simple executive to which you can type the name of the program you want to boot into your machine. Typing "?" will tell you what is available. If you type the name of a program that should exist, and the screen flashes in discontent, it just means that you had the bad luck to contact a boot server without the complete set of programs. If you just type carriage return again, the NetExec will try to talk to a *different* boot server, and that might cure the problem. The most frequently used facilities are the Scavenger, Chat, CopyDisk, and FTP. You can also get a variety of diagnostic programs, the most popular being DMT. And you can get an assortment of games. Warning: many of the games use special Alto microcode, and hence will not run on Dolphins and Dorados.

It is a little known fact that typing shift-swat to DMT will boot the NetExec. Then, typing "BootDP0" to the NetExec will boot your machine. And typing "NetExec" to the standard Alto Executive will boot the NetExec. Knowing all this, you can manage to avoid hitting the actual physical boot button almost entirely, if such should be your wish.

The Alto file system

Most general-purpose computer systems have some sort of file system, and no two of them are exactly alike. Programmers tend to assimilate the assumptions of their local file system so completely that they forget that other systems do things differently. As a result, they sometimes get burned when they start programming under a different system. Let's consider some of the implicit assumptions behind the Alto file system organization.

Alto disks are self-contained. Exception: there exist Altos with two disk drives that can be configured to spread the file system across both disks, as mentioned above. Each one has a single directory in which the visible names of the files are stored. Fine point: multiple directories are permitted, but most software can't handle them. Names consist of a file name proper, optionally followed by a period and an extension. Actually, file names are terminated by a "." and may contain any number of embedded "."s. Dividing the name into two parts with a single "." is purely a convention, though a widely-observed one. Certain conventional extensions exist, e.g., ".mesa" for Mesa source programs. All of this is probably familiar to you from other systems.

Wrinkle #1: The Alto file system supports version numbers that are essentially the same as those of TENEX [9], but almost no one uses them. If you are thinking about using version numbers, don't. There are some lurking bugs in the Alto OS related to version-numbered files.

Wrinkle #2: Because multiple versions are impractical, 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. (However, there is an important exception: Bravo maintains explicit backup files even when version numbers are disabled.)

Wrinkle #3: Alto files consist of pages. Each page carries with it the number of significant bytes it contains. Thus, *in principle*, a file need not be a sequence of full pages followed by a single partially full page. *In fact*, however, strange things will happen if you manage to construct a file in which any page (except the last one) is not full. Fortunately, it is hard to do so.

Wrinkle #4: Alto files always have a page at the front called a *leader* page, which holds various interesting and useful data about the file (e.g., when it was last written). For no good reason, the Alto file system prefers that the *last* page of the file contain less than 512 bytes of data. This means that a logically empty file actually takes two pages, one for the leader and one containing 0 significant bytes of data.

Wrinkle #5: The Alto disk architecture permits 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/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 of the file) into the label field. Now, when the software goes to write a sector, it typically asks the hardware to compare the header and label fields 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. The label field also contains links (disk addresses) to the predecessor and successor blocks of the file. It happens that if the compare logic of the disk microcode sees a particular pattern in a memory word,

it omits the comparison on that word and instead overwrites the pattern with the data from the corresponding disk word. Thus, by cleverly arranging the memory "image" of a label field to be compared, the software can get the safety check on the block identification AND obtain the disk addresses of the neighboring blocks in the same operation. Cute, huh? More information about the disk system and how the software exploits it may be found in {1} and the "Disks and BFS" section of {8}.

You should also know about the Scavenger, a program that rebuilds the file structure (but not the file content) of an Alto disk. Despite the checks and balances of the file system, occasionally things get smashed or lost. When they do, running the Scavenger is the best first attempt to recover them (assuming that your hardware is functioning correctly; running the Scavenger won't help much and could hurt if it is really your disk drive or Alto hardware that is busted). The Scavenger is available from the NetExec, so even if your disk is so messed up that you can't boot it, help is available. You can read more about the Scavenger and what it can do in {10} and [13].

Big Fish in a Small Pond—the Alto disk

Lots of smart people have spent lots of time producing lots of nifty software for the Alto. Programs to manipulate directories, programs to format documents, programs to make pictures and illustrations, programs to transfer files, disks, and messages around, programs to help you write programs, programs to ... oops, you just ran out of disk space.

We all know that you can't have your cake and eat it too. With a small disk, you can't even *have* it, most of the time. Most people are amazed to learn that an Alto disk will hold over 4800 pages (actually, 4872 pages, a little under 2.5 MBytes)—they rarely see one with more than a few hundred available. You will quickly discover that Alto programmers spend a significant fraction of their day switching disk packs or running FTP. Some do both. Of course, users of shared Dorado partitions have to use FTP or its equivalent quite a bit as well, but for different reasons. There's no real cure for this disease, but by being aware of what is costly in space, you can make the pain less acute. Let's see where the space goes by "building a disk" from scratch, so to speak.

Naturally, there are a few things that you just can't live without. You must have an Operating System, an Executive, and a number of files that go along with them. You also need the basic file system machinery (the directory and the disk descriptor so you can allocate disk space). You also should have Swatee and probably Swat, though the latter isn't really essential. There are also a number of small files that the OS and the Executive expect to find around—don't try to get rid of them, they will just come right back. Even so, a disk with just the OS, Executive, Swatee, Swat, and friends still has about 3900 pages free.

Next come the facilities you nearly always want: FTP and Bravo. FTP can always be obtained from the NetExec, but that takes a while and you use it frequently, so most people keep it around on the disk. It consumes about 180 pages. For wizards and hackers, there is a version of FTP hiding inside SYS.BOOT, and one can put together a small kludge that transfers control to it. This way you can have FTP without giving up the full 180 pages it normally needs. Bravo is a good deal larger, weighing in at about 650 pages by the time you count all its related files and a font or two. With FTP and Bravo, your disk is now down to around 3000 free pages.

At this point, things begin to diverge, depending on your plans for the disk. As an example, let's consider a Non-Programmer's Disk. This is a disk designed mostly for producing memos and documents and contains lots of files appropriate for these tasks. When you get it, it has about 1600 free pages—a comfortable amount, but quite a drop from 3000. Where did those pages go? Well,

first there are the forms—files containing templates for things you commonly produce, like memos, letters, etc. The standard bunch only consumes about 40 pages—cheap. Then come the fonts ... and disk space starts to disappear. Screen fonts (so that Bravo can display things) occupy about 100 pages, and you should probably retain the standard bunch (on a Non-Programmer's disk, at least). The real space hogs are subsystems you rarely use. Chat takes 130 pages, and unless you spend a great deal of time talking to Maxc (in which case you probably don't care about Alto disk space), you might as well get it from the NetExec when you need it. Markup and PressEdit occupy another 210 pages, Draw takes 150 more. Unless you use these facilities regularly, you are wasting 300 pages. Neptune takes up some more, unless you feel that you can get by with the standard Executive file system commands. There is also an Executive command "FileStat" that will tell you how big files are. Finally, a Non-Programmer's disk also has BravoBug and some sample documents and illustrations, which together occupy about 100 pages. They are pure fat—you almost never use them, and can retrieve them from appropriate servers when necessary.

Perhaps this tirade on disk space seems superfluous—after all, a Non-Programmer's disk *does* have over 1600 free pages even with all this junk. True, but other disks are rarely so empty. Things get particularly tight on Mesa disks, so it is useful to know just what can be deleted and what can't. The preceding discussion gives you an introduction, but Mesa programmers have been known to go out much further on the limb in their quest for breathing space (e.g., deleting Swat, SYS.ERRORS, and related files that are only needed when the Alto is trying to tell you about a weird error condition). The moral is: know what's on your disk and why it's there. Delete *$, Scratch*, and *.scratch occasionally—it's amazing what you find lying around.

In summary, then, we can categorize some commonly encountered files as follows: Files on the same line generally assume or require that their brothers and sisters exist.

Essential files:
Sys.boot SysDir DiskDescriptor Sysfont.al Swatee
Executive.run Com.cm Rem.cm User.cm

Highly desirable:
Sys.errors Swat
FTP.run FTP.log
Bravo.run Bravo.error Bravo.messages (and various scratch files)

Useful:
Empress.run Fonts.widths
Neptune.run
Laurel.run or Laurel.Image on Mesa disks
various .al font files

Partitioning up the pie: Dolphin and Dorado disks

Users of Dolphins and Dorados don't have to worry quite so intently about disk space, since their local disks are much bigger. In fact, it turns out that their local disks are so big that the Alto OS cannot address the entire disk! Hence, when a Dolphin or Dorado is emulating an Alto, its local disk is split up into separate worlds called *partitions*. Dolphins disks hold two partitions, while Dorado disks hold five; and each partition is 22,736 Alto pages in length (Oh boy, space to burn! Or so it seems for a while...). 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 logical volumes can be bigger than a partition, and frequently are.

Hook, Line, and Sinker

How do you reel in those big fish we've just been talking about? With FTP. After Bravo, FTP is probably the most heavily used Alto subsystem, so it is well worth your while to learn something about its facilities. However, the documentation in [13] is sufficiently old that you should read the relevant section of {10} instead. We'll touch on the high points here.

Full-blown FTP (there are half-blown versions) operates three windows on the Alto display, of which two are interesting. The so-called "user" window is where you conduct your file transactions with another machine, often (but not necessarily) a file server. The "Telnet" window provides you with a stripped-down version of Chat, and is handy when you want to do something that just isn't covered by the file transfer protocol. Most of the time, however, the user window is all you ever look at (or through). There are commands to establish and destroy connections, to retrieve, store, delete, and rename files, and to interrogate directory contents and storage resources.

Much of the flexibility of FTP is derived from its command line processor, which, in conjunction with the Executive's file name completion and *-expansion, provides considerable flexibility and power. With flexibility comes the ability to screw yourself with ease, so FTP implements a few checks that prevent you from doing stupid things, at least without confirmation. You should read about the /A and /> switches remembering that, unless you can afford to maintain multiple versions, once you write on an Alto file, it's gone.

Editing and Producing Documents

In the outside world, document production systems are usually de-coupled from text editors. One normally takes the text 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 a modest step away from the programming paradigm for document production. A single program provides the usual editing functions *and* a reasonable collection of formatting tools. You can't program it 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 printer 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 is a success because most of these issues are resolved more or less by fiat—someone has prepared a collection of configuration parameters (in user.cm) and a set of forms (on a Non-Programmer's disk and the MAXC <Forms> directory) that accommodate 99% of the document production you have to do. Most of the configuration options aren't even documented, so it is hard to get enough rope to hang yourself. If you feel suicidal, there are always wizards about who can answer your every question about Bravo esoterica. The net effect is that you spend much more time composing and much less time compiling.

No one believes Bravo is the ideal solution; indeed, it has a lot of shortcomings that become evident as you begin to push on it. Nevertheless, it is a sufficiently large step forward that you will wonder how you tolerated the old way of doing things. (If this isn't obvious to you after reading [12] and [13], wait until you've used it for a few weeks.) You will also find that the availability of multiple fonts, paragraphing, automatic indentation, and other formatting facilities *inside* the text editor leads you to make prettier programs as well. It just isn't that much more work to create and maintain attractive source text, and a simple set of formatting conventions can be a more potent program documentation aid than comments (see [11] for some examples). There are some operational annoyances with using Bravo formatting, however. The only program which can interpret Bravo formatting information and produce corresponding hardcopy is Bravo itself, and it can only do so on one file at a time and rather slowly. Empress is much faster, but can only handle pre-formatted Press files or simple text (e.g., a sequence of ASCII characters). There is a Hardcopy subsystem that takes a list of files and feeds them one-by-one to Bravo for hardcopying (it uses a Bravo macro [13] to eliminate manual intervention), but this is a kludge at best. Therefore, some people feel that Bravo formatting is just too much trouble and instead do it "by hand". They are a small minority.

When Bravo crashes

Like all text editors, Bravo breaks once in a while. There is nothing quite like the sinking feeling you get when a large number of your precious keystrokes gurgle away down the drain. When they do, you probably have an instinctive response (conditioned by previous editors you have used) to run the editor again to find out what state your file is in. *Resist this impulse at all costs*—it is the *worst* thing you can do.

Bravo has a "replay" mechanism, meaning that it records all of its actions in a file and is capable of replaying an editing session (yes, even one involving multiple files) from the beginning. *However*, all replay information is thrown away when Bravo initializes *unless* you tell it that you wish to replay the immediately preceding session. If Bravo crashes on you, by diving into Swat or displaying "bootlights", your best bet is to re-boot your Alto, use FTP to obtain BravoBug (unless, of course, you already have it), refresh your memory about how replays work [13], then run BravoBug. Bravo/r *is not an acceptable substitute, despite a popular rumor to that effect!* More details are available in [13]. The essential notion is that you must not run Bravo in the usual way, or you will forfeit your opportunity to do a replay.

Square Pegs in Rhombic Holes—Alto Mesa

For years BCPL was the only implementation language for the Alto. Naturally, a nice cozy environment for BCPL programs (and programmers) gradually developed, and the *cognoscenti* could guess how subsystems would behave in unusual cases because they knew that the programs operated in this environment. Then, along came Mesa and the end of innocence. Mesa programs either have to mimic the behavior of the BCPL environment in situations where they supply overlapping function, or risk being branded "incompatible".

Okay, so that's a bit melodramatic. Nevertheless, Mesa *is* faced with the problem of adapting to an

environment that it finds less than ideal. As subsystems coded in Mesa begin to emerge, subtle incompatibilities appear (e.g., Mesa's inability, at present, to support version-numbered files). Mesa's more modern approach to memory management (implicit segment swapping instead of explicit overlays) has the disadvantage of consuming considerably more disk space, largely because it has become much easier to ignore the constraints imposed by the Alto's small primary memory.

Mesa is nevertheless *the* programming language for successors to the Alto. These machines have an architecture designed to support Mesa comfortably, and BCPL will fade away now that these machines have arrived on the scene. Alto Mesa may be likened to a size 12 foot in a size 11 shoe; Pilot Mesa is _____ (you fill in the blank).

Smalltalk

[This section was contributed by John Shoch.]

Smalltalk is both a programming language and a programming environment, developed by the Learning Research Group with lots of help from other folks in CSL and SSL. The system has always been intended to serve as both a powerful language for use by experienced programmers and an easy language to be learned by children; some of the work of LRG has been aimed at testing out these systems with kids.

As a programming language, Smalltalk is an "object-oriented" system which provides a uniform epistemology:

* Every "object" is an "instance" of some "class".
* The class definition describes the behavior of all its instances.
* Objects communicate by sending messages.

(Geneologists will recognize major parts of Simula and Lisp in our bloodline, combined with traces of many other languages.)

But Smalltalk is more than just a language design—it is a highly interactive, integrated system which tries to merge together many functions that are often viewed as separate subsystems: writing programs, editing text, drawing, real time animation, generation of music, and more. This view meshes well with the notion of a small, single-user personal computer (the "Dynabook"). Currently, a large project is under way to publish the details of Smalltalk 80, and to make it available at a very low level to outside manufacturers of small computers, in the hope of getting it out into the world by that route.

There have been many different releases of Smalltalk, but there have been three principal designs so far:

1) Smalltalk 72, a fully interpreted version developed for the Alto, and used in some of the original work with kids.

2) Smalltalk 76, a newer version incorporating the design of a virtual Smalltalk machine, a microcoded version of this on the Alto, a compiler to produce byte codes executed by the virtual machine, and an object-oriented virtual memory (called OOZE) upon which the whole thing sits.

3) Smalltalk 80, a still newer version, currently being implemented, with a Large Object

Oriented virtual Memory (called LOOM).

For more information, take a look at [22] and [23]. The "Smalltalk 72 Manual" is now both out of print and out of date, but did provide lots of interesting examples and discussion; try to borrow a copy from someone.

No Computer Scientist is an Island: the Laurel message system

We rely very heavily on an electronic mail system. Since people spend much of the day at their Altos, notices posted on a central bulletin board are not likely to be seen rapidly. Accordingly, most announcements are broadcast (to expansive distribution lists) using our electronic mail system. 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 should attend".

We also use the electronic mail system as a way of recording the progress of working groups and projects. Minutes of meetings, design documents, and related materials often pass as messages among group members. A file in which a copy of each such message is retained becomes a valuable archive of the project history and is quite painless to maintain. Many individuals keep archival files of their messages as well.

In the bad old days, the only generally available facility for sending messages was a Maxc subsystem called SNDMSG. A separate program, MSG, was commonly used to inspect and classify incoming messages. Consequently, people who had no other reason to use Maxc were compelled to process their mail there.

Our replacement for MSG is an Alto-based message system named Laurel; Laurel is the message composing and examining program, while Grapevine is the distributed transport mechanism, a collection of server machines that conspire to deliver mail. Because Laurel stores mail locally on the Alto disk and requires a moderate amount of disk space (about 500 pages), most users find it necessary to dedicate a disk on which they process all their mail. If you like to read your mail frequently and you don't have a double-disk system, you find yourself switching disks a lot. The latest version of Laurel (Laurel 6) has lots of fun new bells and whistles that are worth getting to know: check out the modeless editor, thought by many to be the greatest thing since sliced bread, and try "Run"ning some ".laurel" programs in the bottom window. Now that Laurel 6 is around, you can get the facilities of Chat, Ftp, and the Alto Executive all without leaving the Laurel cocoon.

A Printing Discussion

You might expect that Xerox Corporation might be more than a little interested in printing. Indeed, we are so interested that we have created an array of printing facilities sufficient to confuse any newcomer. Let's try to understand the basics.

First of all, there is an important difference between *copiers* and *printers*. A copier obtains its input by scanning a physical image in some way. A printer obtains its input image in digital form from some external source, e.g., an interface to an Alto.

There are a lot of printing programs about: Press, Spruce, Empress. There are a lot of printers too: Dover, Sequoia, Versatec. To make matters worse, each of the *instances* of each of these printers has a name as well: Clover, Menlo, Lilac, Daisy. As you might expect, not all programs can talk to all printers, but we're working on that (see axiom 2 below). Here are a few axioms that may help you reason logically about all this:

1) There are no line printers around here. All of our printers are built on top of Xerox copier printing engines that have been lobotomized and brainwashed to understand the babbling of an Alto instead of an optical input scanner.

2) Press files are the Esperanto of documentation. Most printer servers demand that the documents you send them be in Press file format. This means you have to convert whatever you have in hand (usually text) to Press format before a server will deign to print it. There are several ways to do this.

3) Press files are hairy. Some printer servers don't support the full generality available in a Press file. Generally, however, such servers will simply ignore what they can't figure out, so you can safely send them any Press file you happen to have.

4) There is an extensive collection of standard fonts, and they are mostly straightforward to use. Be prepared for a few surprises if you insist on building your own. The most unpleasant surprise is that you have to be a wizard in order to print with your 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 in anyone's estimation.

In general, if you simply want to make a memo, or a listing of a program, or a copy of a documentation file that someone has sent you, things are quite straightforward. Bravo's hardcopy command [13] will take the file you are editing, convert it to Press format (including all the formatting information you have supplied), and ship it directly to any printing server you specify. The important server program to know about is Spruce, which understands everything Bravo can produce. Spruce is the current driving program of choice for Dovers and Penguins, and is the server you will use for almost everything unless you are a graphics hacker.

Spruce will accept Press files from any source (though it does not implement all Press features). Standard documents and memos are typically stored in Press format, so you can ship them directly to your favorite Spruce server. From an Alto, use the Empress program; from Maxc or an IFS, use the PRESS command.

Empress can tell the difference between a Press file and a text file, and will convert a text file to Press format if necessary before sending it to the printing server. If you do this a lot, you will want

to know about various options that apply to this conversion—see {10}. In such cases, Empress uses a single font, which generally has a fixed pitch. This is the way we simulate a line printer.

Spruce servers have a collection of fonts stored locally. Press files do not contain the representation of the fonts they require, only their names. Naturally, if a Press file is produced using fonts that a Spruce server doesn't have, the server will have a hard time printing it. Spruce will attempt a reasonable substitution for unavailable fonts, and tell you about it on the break page of your listing. If you have chronic font difficulties of this sort, contact your local Spruce maintainer.

Most frequently traveled paths through the printing maze

Running on	*Input file format*	*Output desired*	*Program to use*
Maxc	Text	Press file/printer	PRESS command
	Press	Press printer	PRESS command
IFS	Press	Press printer	Press command
Alto	Text	Press file/printer	Empress, Bravo
	Bravo	Press file/printer	Bravo
	Press	Press printer	Empress
	Draw	Press file	Draw or ReDraw
	SIL	Press file	SIL

A few caveats go along with this table. First, it is typically easier to format and print large files from Maxc (because of disk space considerations) than from an Alto, but it often takes longer. Second, you should know that these various programs have a large number of options and defaults, and they are not always consistent. Beware of printing groups of files with *-expansion on Maxc (particularly *.*) unless you are certain you are doing it properly. (LIST *.* is a *disaster*, for more reasons than you might think.) For more details about printing, and before you try to do anything clever, read {21}.

Beyond the Black and White Horizon - MAXC and the Arpanet

Sitting at your Alto, you can easily forget about the other computing facilities that are within your grasp. One important server on the network is MAXC, which is a home-grown microprogrammed processor masquerading as a PDP-10, and running Tenex. Maxc is connected to the Arpanet, so it is possible for you to reach out to any machine connected to the Arpanet, at least in principle. In practice, not many people do (and there are restrictions imposed by our ARPA contract as well), except to send messages to people at other Arpanet sites. Laurel understands Arpanet names in messages, so you don't need to use Maxc directly to send or receive Arpanet mail.

Looking under Rocks

All Alto users should know about various interesting files and directories. There is no coherent logic to the placement of "general interest" files and directories, nor even to the division between Maxc and Ivy. Browse through the glossary at the end of this document to get a rough idea of what's around. If something is 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.

Browsing on Maxc

The primary directories for documentation on Maxc are <AltoDocs>, <Doc>, <PrintingDocs>, <Mesa-Doc>. As you can see, the naming conventions aren't very consistent, so you may have to fumble around a bit before you find the right one. The Tenex file name completer (ESC) can take some of the difficulty out of remembering, as can a quick glance in the glossary at the end of this document.

Just because you don't find a particular file, don't give up! Tenex has an automatic facility called "archiving", which moves infrequently accessed files to tape. It sometimes happens that the documentation you are looking for has been archived. There is a Tenex command, "Interrogate", that will help you locate an archived file—see [9].

Maxc is still the file repository "of record" for some families of files, though that burden is gradually being shifted to various IFS servers. These servers frequently have duplicate copies of documentation, packages, subsystems, and the like. This is done partly for redundancy and partly to decrease the load on Maxc when a new version of a popular Alto facility is released. Duplicated directories always have the same name on IFS servers, but are not always scrupulously maintained, and therefore may be inconsistent, incomplete, or obsolete. Proceed with caution.

Browsing on IFS servers

IFS servers don't have an archiving facility (yet, although they may soon), which means that you are less likely to overlook something interesting. IFS supplies a general sub-directory structure which the Maxc file system lacks, and as a result there are many more pigeonholes in which to look. For example, on Maxc you might look for

<AltoDocs>MyFavoritePackage.press

while on IFS you would probably look for

<Packages>Doc>MyFavoritePackage.press
<Packages>MyFavoritePackage>Documentation.press

or perhaps some other permutation. This requires a bit of creativity and a little practice. However, if you use the "Chat Executive" and 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 (and expect such searches 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, an asterisk can *only* be used to

wildcard either the entire filename or the entire extension. If you want to refer to all of the files on a TENEX directory, you must say "*.*", not just "*"; this lack of "forward compatibility" (the opposite of backward compatibility?) has tripped up many a searcher.

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 on 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."

Mostly 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 used to be 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.

"Hey guys, up-level!"

The conversation has degenerated to a discussion of nitty-gritty details. This phrase is often preceded or followed by: "We're having a printing discussion."

... smashed to zero

A quaint way of saying that some memory location has acquired the value zero when it should have something else. "Smashed" is much preferred to "clobbered" in local argot, though in this context it seems about as appropriate as using a wrecking ball to stack bricks.

"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.

Some CSL Lore

Here are a few bits of information specific to CSL that you should know:

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 Computing Forum on Thursday afternoons, which is not specifically a CSL function and is open to all Xerox employees. Dealers are also used for announcements which 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), with Mother Xerox picking up the tab.

The CSL Archives (not to be confused with TENEX archives) 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, but they make interesting reading for anyone curious about the history of any particular project. You will find it most informative to browse the archives from time to time, just to see what's been going on in those projects you just haven't quite had the time to monitor. Ask someone to point you at the cubicle where the archives are stored.

If you are a CSL member and need a new disk pack, see Mike Overton in the CSL lab (across from the Commons).

Some ISL Lore

Here is one item of information specific to **ISL**:

ISL also has a weekly meeting, on Tuesday's starting at 11:00 am. This meeting is still nameless. In fact, Chuck Geschke has a standing offer of a bottle of fine Cabernet to anyone that can come up with a name for this meeting that Chuck likes enough to adopt: the Cabernet is aging unclaimed as of this writing.

Gracious Living Hints

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; 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 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; Dolphins and Dorados are another matter, because of the problems of sharing disk space. But, if you borrow someone's Alto, or use their office for some other 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 owner's disks back in her machine, 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 Dorados, and Sharing Local Disk Space

The sharing of disk space has its own ethics and morality. First of all, don't attempt to "share" a private partition, except with the explicit prior consent of the owner; only public partitions are intended to be generally used in shared mode. The law regarding public partitions is that everyone is supposed to be living cleanly: that is, completely backed up on remote file servers. Hence, according to the "letter of the law", you are completely within your rights if you delete anything from a public partition, whether you really need more space or not. And in fact, even if the "thing" that you are deleting is a standard system facility, such as the Mesa compiler. But going along with the "letter of the law" is the "spirit of the law". We can all use public partitions more effectively and spend less time in CopyDisk and FTP if we treat the public partitions with some care. Try not to delete standard systems, unless you really need the space. And be even more careful not to leave non-standard versions of standard files on a public partition, where they might confuse the user who follows you. Clean up after yourself if that is convenient, to give the next user a pleasant start at her Dorado time.

In general, be aware that it is only good citizenship to try to get a sense of the common Dorado usage patterns in your environment, and not to upset those patterns when there are clear alternatives. Oh boy, do I feel old! When I first started hanging around CSL, the term "citizenship" referred to a score computed from the timing and extent of one's usage of cycles on Maxc. And a big list giving everyone's citizenship rating as well of lots of other Maxc statistics was posted each month near the coffee alcove. Nowadays, right across the hall, you will find the sign-up lists for the pool Dorados of CSL and ISL. Right next to the lists, you will find posted a full list of the current rules governing signing up, and a map showing the locations of the Dorado

terminals.

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 are right underneath 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", which is the name of someone who hasn't logged in to their Dorado partition. 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, (?) leave it on top of Clover, or (?) stick it back in the output hopper.

The phone system

If you make a significant number of personal long-distance phone calls off of 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 moral: if you think you aren't being paid enough, you should start agitating for a raise. Furthermore, if enough suspicious calls are made without restitution, PARC (being a bureaucracy) will impose some bureaucratic "solution" on us all.

But enough of preaching: so as not to end on a sour note, let's finish up by discussing how the phone system works, anyway. The offices within PARC have four-digit extensions within the 494 exchange (what Ma Bell calls 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 [Oh, come now, surely you know about phone books!] for more details. Dialing a single 8 gives you different sounding dial tone, and puts you onto the IntelNet (not to be confused with the InterNet, of course). 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 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 and go via Ma Bell from the beginning.

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.

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. 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 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 a third time will flush the annoying ringing or busy signal.

A Glossary of Terms, Subsystems, Directories, and Files

(and acronyms, protocols, and other trivia)

ADL keyboard ADL is a acronym for the Advanced Development Laboratory, a part of **PARC** located in Southern California. This organization came up with the **ADL keyboard** as an inexpensive alternative to **Microswitch keyboards**, and they were used on the first few builds of **Alto** II's. They have extra columns of function keys separated from the primary keys on both sides; the feel of an ADL keyboard is unique.

Adobe A program for submitting, collecting, and managing AR's. Subsumes the functionality of the former **ARSubmit.**

Alpine A project within **CSL** to build a transactional file server for use by data base systems to be built within **Cedar** and the hypothetical **Cedar** universal file system. A follow-on to **Juniper.**

Alto If you don't know by now...

AltoFontGuide.Press A file that tells all about the existing families of display screen raster fonts, and describes how they are organized as different subdirectories on **[Ivy]<AltoFonts>.**

APC In the armed forces, an acronym for Armored Personnel Carrier; used to refer to the massive cabinet designed to housebreak a **Dorado**. No one seems to know what the acronym really stands for in the **Dorado** case: Armored Personal Computer, or Armored Processor Carrier are two possibilities.

AR Acronym for Action Request: a report of a bug or a request for a new feature. **AR**'s are part of mechanism developed within **SDD** for handling feedback from users of programs to their implementors. **AR**'s are only relevant for software produced by **SDD.**

ARSubmit Network-bootable program for submitting **AR**'s (widely reviled).

ASD Acronym for Advanced Systems Department, which was once a part of **XBS**, but was disbanded some time ago.

bank A unit of measurement of primary storage, equal to 64K 16-bits words, or equivalently, 128K bytes. An **Alto** II has four banks, while **Dorados** 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.

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

BCPL A system programming language used as the basis for many Alto facilities. Also, the compiler for that language.

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

BITBLT (pronounced "bit-blit"). A complex Alto 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 Alto display hardware and microcode process what is essentially a bitmap of the image to be displayed. At PARC, bitmaps are normally stored in word-aligned, pure row-major order.

Boardwalk Another name for **Building 35,** the main building of **PARC** (mostly heard in the conversation of residents of **PARC-place).**

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 an Alto keyboard used (sometimes in conjunction with the keyboard) to **boot** some program into execution. One of several such small buttons on a **Dolphin** or **Dorado.**

boot server A computer on the network that provides a retrieval service for certain stand-alone programs. See **NetExec.**

bootlights A screen pattern resembling a city skyline. Occurs occasionally when some erroneous unanticipated condition arises, e.g., getting a parity error in a **BCPL** program on a disk that doesn't have **Swat.**

Bravo.run An integrated text editor and document formatting program that runs on the Alto; a vital program that nevertheless is no longer maintained or supported.

BravoBug.run A program used when Bravo crashes to **replay** the editing actions up to the point of the crash, and/or to *report* the problem. (For some time, Bravo has been receiving only such maintenance as is truly unavoidable, so ignore the reporting function.)

BravoX.run 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.

bug award Refers to a relatively recent 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 (the rough equivalent of a gold star).

Building 32 A part of **PARC,** located on Hanover Street, north of Page Mill. Also called **PARC-place.**

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. Occasionally called **Boardwalk** (to contrast it with **PARC-place**).

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

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

Butte A new 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.

byte code **Lisp, Mesa, 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 codes** are in turn interpreted by special microcode.

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

Cascade See **PreCascade.**

Cedar A large project in CSL to build a programming environment for essentially all of CSL's future applications. **Cedar** is also the name of the programming language upon which this **Cedar** system is built, a variant of **Mesa** augmented by garbage collection and run-time types. The design of the **Cedar** environment was strongly influenced by the programming environment and services in **Interlisp.**

CedarGraphics A subroutine package of graphic primitives that forms an important part of **Cedar.** Its design was heavily influenced by the results of experimental systems written in **JaM.**

CenterPunch An old name for **Hornet.**

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

Chat A **subsystem** that permits teletype-like, interactive access to a remote computer on the network. Used also to refer to a facility resembling that provided by this subsystem, e.g., **FTP** is said to have a Chat window. Chat is mainly used to communicate with **Maxc** and **IFS** servers.

Chipmunk A **D-machine** 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 process control program being written for **ICL** with help from **CSL.**

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.

client A program (rather than 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 that lists by family name, face, size, and rotation all of the fonts in **Clover**'s font dictionary; available on **[Ivy]<Fonts>.** To see the characters themselves, check out the book located on top of **Clover** called CloverCharacters.Press.

CoCoPilot A **world-swap** debugger for **CoPilot.**

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

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

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

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

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

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

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

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

Dandelion The (former?) 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".

DDS.run Acronym for Descriptive Directory System. An Alto **subsystem** providing more sophisticated manipulation of the file system than is available with the **Executive.** See also **Neptune,** which is a poor man's DDS: it provides most of the functionality but uses less disk space and time.

DfFiles A collection of programs for describing and automatically retrieving and building complicated **Mesa** systems. Built by Eric Schmidt as a forerunner of real **system models, DfFiles** primarily addresses the problems engendered by our current feudal and highly non-universal collection of file systems. See

[Ivy]<CedarLib>DfFiles>DfFiles.Bravo or ditto.Press.

dirtball A small, perhaps struggling outsider; not in the major or even the minor leagues. For example, "Xerox is not a **dirtball** company". The author is uncertain to what degree this term is derogatory.

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

DLISP A version of **Interlisp** running on Maxc that communicates with some fancy display manipulation facilities on the Alto; largely supplanted and not to be confused with **Interlisp D.**

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. 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.

DOC A **Cedar** concept and collection of programs for giving flexible active views of various forms of structured data, text or otherwise. **DOC**s were once called **documents.**

Dolphin Generic name for a personal computer once called the **D0.**

Dorado Generic name for a high-performance personal computer designed by **CSL.**

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 **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.run An Alto **subsystem** that permits interactive construction of pictures composed of lines, curves, and text; not a very solid system. **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.**

Dumper.boot A file used for desperation debugging. 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 make LISP do what you mean, not what you say.

EARS 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!)

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.run An Alto **subsystem** used to convert text files to **Press** format and ship them to a **Press printer server.**

EOS Acronym for Electro-Optical Systems; a part of Xerox located in Pasadena.

Ernestine An Alto server that allows you to dial up and read your mail while it is sitting in your **Grapevine** mailbox; primarily useful when, due to travel or whatever, your personal computer and **Laurel** are unavailable.

Ethernet The communication line connecting several Altos (or other 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 **Gateways** (which is more correctly called an **InterNet**). Every Ethernet within an **InterNet** has a unique identifying number. 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. All of the Ethernets currently within PARC are of the earlier, 3 MBits/sec variety.

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

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 defaulting of extensions. Some commonly encountered extensions are:

~	an Executive command (not really an extension)
al	Alto screen font
bcd	Mesa object program module
bcpl	BCPL source program module
bfs	an entire 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 config including path names for files
dm	dump file (i.e., several logical files stored as one)
error(s)	Swat error message file
image	executable (Mesa) program
ks	new format Alto screen font
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
press	Press file
run	executable (BCPL) program,
sil	SIL source file for a drawing
st	Smalltalk source program text
symbols	Mesa symbol table (for debugging)
syms	BCPL symbol table (for debugging)

tex	TEX source text
tfm	font metric information, for use with TEX
ts, typescript	typescript file (log of interaction)

file server A computer on the network that provides a file storage and retrieval service. **MAXC**, **IFS**, and **Juniper** are three different types of file servers, though they provide related facilities.

FLG (pronounced "flug"). A switch (usually in Lisp programs) 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 for understanding.

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

Fonts.widths A file containing character-width information for a large number of fonts. Used by many programs that do text formatting while producing **Press** files. The standard source is **[Ivy]<Fonts>Fonts.Widths.**

FTP.run Acronym for File Transfer Protocol (or Program). An Alto 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 **Ethernets.** Gateways may also perform certain server functions, such as **name lookup.**

Grapevine The distributed electronic message transport system; it has a set of protocols all its own, and provides various server functions such as authentication.

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-names. 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.**

Guibas A unit of complexity of mathematical arguments, defined to be Leo's maximum sustainable hair-level. For practical purposes, the milliGuibas is a more reasonable unit.

Hardy A **Tool** that provides the functionality of **Laurel** in the **PreCascade** environment: a client of **Grapevine.**

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 An Alto-based program for creating and editing integrated circuit designs graphically and interactively.

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 the **Science Center** of **PARC**, located in **Building 34.**

IFS Acronym for Interim File System. An Alto-based file server. Several distinct IFS servers exist on various **Ethernets**, including **Ivy, Ibis, Iris, Idun, Igor, Phylum, XEOS,** Erie, and about 15 others.

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.*

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

Inscripts A mechanism for keeping track of user input to a program in a general way (key strokes, mouse clicks, and the like), for use 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.

Interlisp A dialect of Lisp with a large library of facilities and a 15-pound reference manual. **Interlisp D** is a dialect of **Interlisp** that is being developed for use on **D-machines** by people at **PARC-place.**

InterPress A printing file format standard that is being developed: a second cut at the same issues addressed by **Press** format.

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

Ivy An IFS server in **PARC**, used by **CSL** and **ISL.**

[Ivy]<AltoFonts> A directory on which screen fonts for the Alto are stored (extension .AL). Subdirectories are used on this directory to distinguish various families of display screen fonts that have accumulated over the years.

[Ivy]<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 pack is to use **CopyDisk** to retrieve one of these disk images.

[Ivy]<Cedar> The source of actual **Cedar** code and documentation.

[Ivy]<CedarLib> A library of packages for use within **Cedar.**

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

[Ivy]<Mesa> A directory on which **Mesa** programs (source and object) and documentation are stored. Additional facilities of interest to Mesa programmers may also be found on **<MesaLib>**.

[Ivy]<MesaLib> A directory on which **Mesa** utilities and packages (source, object and documentation) are stored. Standard Mesa programming facilities may be found on **<Mesa>**.

[Ivy]<Pilot>Doc> The home of the memo **SettingUpPilot.**

[Ivy]<APilot> A directory that is periodically reinitialized as a complete copy of [Idun]<APilot#0>, where "#" is the most recent pilot release (currently #=6).

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

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

Juniper An Alto-based distributed file system, built within **CSL.**

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.**

KBA Acronym for Knowledge-Based Assistance. Refers to a former project in **CSL.**

KRL Acronym for Knowledge Representation Language. Refers to a former project in **CSL.**

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

Laurel An Alto-based, display-oriented program that provides access to the facilities of **Grapevine.**

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

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

1 implementors only
2 implementors 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.**

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

LRG Acronym for Learning Research Group, a part of the **Science Center** of **PARC.**

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.

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. For documentation, see [Ivy]<CedarLib>MakeConfig>MakeConfig.Press and ditto.bravo.

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

Markup.run 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). See **[Maxc]<AltoSource>** and **[Maxc]<Alto>** for corresponding source and object files.

[Maxc]<AltoSource> A directory on which source versions of standard Alto programs are stored. Corresponding object versions and documentation are stored on **<Alto>** and **<AltoDocs>**, respectively.

[Maxc]<Forms> 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 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]<Secretary> A directory containing standard distribution lists for use with **SNDMSG.**

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

Mokelumne A former release of **Pilot.**

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

menu A collection of text strings 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.**

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 make keyboards. The standard **Alto** keyboard, also in use at **PARC** on **Dolphins** and **Dorados,** is made by Microswitch. In contrast, see **ADL keyboard.**

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.

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.

MType Another early product of the **system modelling** effort: for information, see [Ivy]<CedarLib>MType>MType.Press or ditto.bravo.

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.

Neptune An Alto **subsystem** providing more sophisticated manipulation of the file system than is available with the **Executive.** See also **DDS,** which provides still more bells and whistles at the expense of time and space.

NetExec.boot A mini-**Executive** usable without a disk and obtainable directly from 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.**

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.

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, of which **SDD** is a part.

Orbit A high performance 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.

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

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

PARC Acronym for Palo Alto Research Center.

PARC-place Another name for **Building 32**, located on Hanover.

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** is 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 **pages** (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 A complete description of a directory or subdirectory on which files may be stored—everything you need to know to get the file except the file name. 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, each followed by a right angle bracket.

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. **Penguins** have better **solid-area development** than **Dovers**, and can also print two-sided. They are normally driven with **Spruce.**

Phylum An **IFS** in **PARC-place.**

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. **Pilot** is also the current base for **Cedar.**

Pine The page-level file access protocol used by **Juniper.**

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 An old name for the **MIG.**

Poplar An interactive programming language system running on the Alto, 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 **PreCascade** environment.

PreCascade The current version of an interim integrated **Mesa** development facility based on **Pilot.** A future version (and a prior version, confusingly enough) will be (was) called simply **Cascade.** When operating in **PreCascade,** editing, compiling, binding, and creature comforts all happen inside of the **CoPilot** world. A **world-swap** occurs to the **Pilot** world only when actually trying out the program currently undergoing development.

Press A file format used to encode documents to be transmitted to a printer. Also, a printing server program, written in BCPL, that can print curves and raster images as well as characters and rules.

PressEdit.run 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.

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

printer server A computer that provides printing services, usually for files formatted in a particular way. The term also refers to the specific software that converts such files into a representation that can be processed by a specific printer hardware interface. **Spruce** is an example of a printer server program.

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

800	typewriter-based, word-processing terminal
860	display-based, word-processing terminal
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

9200 offset-quality, .5 sec/page copier
9700 offset-quality, .5 sec/page, laser-scan printer

PSD Acronym for Printing Systems Division.

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".

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

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 end in an "↑", as in "CSL↑.PA".

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

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 will print directly on masks for integrated circuits.

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

Rubicon The current release of **Pilot.**

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 A program available through the **NetExec** that checks for damaged file structures in a **BFS** and tries to repair them.

Science Center Half of PARC; the other half is the **Systems Center.** The rationale behind the specifics of the division are unclear.

scroll Refers to a method of repositioning text on a display as though it were part of a long, continuous sheet of paper.

SDD Acronym for System Development Division, a part of **OPD.**

SettingUpPilot A memo on how to set up a **Pilot** world on a **Dolphin** and **Dorado**, with lots of good dope on what is really going on; available on **[Ivy]<Pilot>Doc>.**

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

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

SLOT Acronym for Scanning Laser Output Transducer.

Smalltalk An integrated programming system based on object style and message passing, invented and developed by **LRG.**

Snapshot Another early product of the **system modelling** effort, now mostly used to clean up **partitions** by means of the "deleteall" option. For more data, see [Ivy]<CedarLib>Snapshot>Shapshot.press or ditto.bravo.

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

Spruce A program that takes certain simple **Press** files (primarily text and **rules**), converts them to a form acceptable by an **Orbit** interface, and prints them.

SSL Acronym for System Science Laboratory, a former part of the **Science Center** of PARC. What used to be SSL now exists as a collection of "groups" or "areas" (of which the author does not have a very good model).

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. The 8000 architecture was created in **CSL.**

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

subdirectory File directories on an **IFS** can be divided into a hierarchical collection of subdirectories. The subdirectory names are listed from the top of the tree down to the bottom, and are separated by the single character ">". For example, the directory **[Ivy]<CedarLib>** has a subdirectory called **MakeConfig** on which files related to the **MakeConfig** program or stored. To refer to the file MakeConfig.Press on this subdirectory, one would write:

[Ivy]<CedarLib>MakeConfig>MakeConfig.Press

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" and "shift" keys to invoke the debugger. 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 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 modelling A part of the **Cedar** project, aiming at giving programmers help is describing the structure of large systems: getting all the versions correct and the like.

Systems Center Half of PARC; the other half is the **Science Center.** The rationale behind the specifics of the division are unclear.

Tajo An environment for developing **Mesa** programs that exists in both **Alto** and **Pilot** versions; this environment comes complete with a consistent philosophy about many things, including how screen space should be used, which window should be listening to the keyboard at any given moment, and much more. Each facility in the **Tajo** environment is called a **Tool**, and **Tajo** itself was once called the **Tools Environment.**

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 two implementations of TEX at **PARC**, one in Sail that runs on **Maxc**, the other in **Mesa** for **D-machines.** TEX can handle mathematical formulas, but doesn't let you see anything like what you get.

thumbing A technique of positioning a file (usually text) to an arbitrary position, usually for viewing on a display.

Tioga A project in **ISL** to build an integrated editor/typesetter in **Mesa.**

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.

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

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

Universal File System At the moment, merely a dream: in particular, the dream that someday all of our files will live in a common system, implemented in a distributed and reliable fashion by remote servers, for which all local disks will merely be automatically-managed caches. There is some hope that this dream will be turned into reality someday by something to be built on top of **Cedar** and **Alpine.**

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 **Bravo.** See **client.**

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

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

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

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. Most of the debuggers in our current environment 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.

XBS Acronym for Xerox Business Systems; a former organization, **XBS** was subsumed by **OPD.**

XEOS An **IFS** located at **EOS.**

XGP 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** located at **PARC-Place.**

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

References

Reference numbers in [square brackets] are for conventional, hardcopy documents. Reference numbers in {curly brackets} are for on-line document files. The notation used for on-line files is: [FileServer]<Directory>SubDirectory>FileName.Extension .

Each reference is followed by a brief description of what you can expect to find in the cited document.

If you can't find some of the on-line files, they may have been archived. See the section on "Looking Under Rocks".

{1} [Maxc]<AltoDocs>AltoHardware.press

[2] Metcalfe, R. M. and Boggs, D. R. **Ethernet: Distributed Packet Switching for Local Computer Networks.** *Communications of the ACM 19*, 7 (July 1976), pp. 395-404. A description of the Ethernet's functional organization, with a discussion of error recovery strategies.

{3} [Maxc]<AltoDocs>NetTopology.press. Contains a picture of the entire internetwork configuration: currently four pages.

{4} [Maxc]<Pup>FTPspec.press. A functional specification of the file transfer protocol, independent of implementation in any particular language or system.

{5} [Maxc]<Pup>EFTPspec.press. A functional specification of the "easy" file transfer protocol.

{6} [Maxc]<Pup>Telnet.press. A functional specification of a protocol for interactive teletype-like communication between computers on the network.

{7} [Maxc]<Pup>MiscServices.press. Describes a variety of simple protocols (usually a single exchange of packets).

{8} [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.

[9] 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.

{10} [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.

[11] 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.

[12] Jerome, Suzan. **Bravo Course Outline.** Xerox PARC Internal Report, undated. Oriented to non-programmers.

[13] **Alto User's Handbook.** Xerox PARC Report, November 1978. An introduction to Alto facilities and reference documentation for several commonly used subsystems, including Bravo, Laurel, FTP, Draw, Markup, and Neptune.

[14] **Sunset Western Garden Book.** Lane Magazine and Book Company, Menlo Park, Ca. The definitive document on Western gardening for non-botanists.

{15} [Maxc]<Pup>GatewayInformation.press. Describes the protocol for obtaining packet routing information from the Gateways.

{16} [Maxc]<Pup>Pup.press. A functional specification of the PUP mechanism for packet-based communication on the network.

{17} [Ivy]<Laurel>Laurel.press. Documentation for the Alto-based, electronic mail system. Also available as blue-and-white report CSL-81-6.

{18} On-line documentation for Smalltalk is (always?) in a state of flux. Consult a member of LRG for a current pointer.

[19] Maybury, W., Mitchell, J. G., and Sweet, R. **Mesa Language Manual.** Xerox PARC Internal Report CSL-79-3, April, 1979. A cross between a tutorial and a reference manual, though much closer to the latter than the former. Details of the Alto implementation appear in other, on-line documentation—look on [Ivy]<Mesa>.

{20} [Maxc]<AltoDocs>AltoUsersPrimer.press. A more complete (and neutral) introduction to Alto-land, intended at least in part for non-programmers.

{21} [Maxc]<PrintingDocs>Printing.press. The "entry document" for printing services on Maxc and the Alto.

[22] Kay, A. C. "Microelectronics and the Personal Computer". *Scientific American*, September, 1977, pp. 230-244.

[23] **Personal Dynamic Media.** Xerox LRG/SSL report 76-1, 1976.

{24} [Ivy]<DoradoDocs>DoradoBooting.press. The operation and mechanisms of booting a Dorado.