A Field Guide to Alto-Land

or

Exploring the Ethernet with Mouse and Keyboard

BY ROY LEVIN

REVISED APRIL 1979

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

c Xerox Corporation 1979

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 (supposedly) you can browse through and read the parts that look interesting. 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. 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-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}.)

A *Rosa* by Any Other Name ...

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 PARC, ASD, 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 taken from the Sunset *Western Garden Book* [14]. Individual Altos also have names which frequently, but not necessarily, come from the same source. These names are chosen by individuals and registered with Art Axelrod. 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.

The Alto hardware

The first thing an immigrant notices about the Alto is that there are a lot of them. It seems that every office has one. In fact, there are order of 1000 Altos in existence, though only some of these are at PARC. The genus 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 keys, and a processor/disk cabinet. You can normally spot an Alto II by its larger keyboard with extra buttons on each side. You can even spot an Alto II blindfolded - the "feel" of its keyboard is unmistakable. In fact, at a major demonstration attended by many important Xerox people, Alto II's were used because of their flexibility, but Alto I keyboards were attached to them. Alot II's are getting cleverer, however; the newer models come with keyboards that "feel" more like Alto I's, but cost a lot less.

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 the standard

> Alto 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, particularly since there are 500 or so Altos around. 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 ROM on an Alto II.

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, horizantal 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 keys 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.

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. 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.

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. 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 (P_arc U_niversal P_ackets) {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:

0) EEFTP - a grand-daddy of file transfer protocols (Experimental Ether FTP). No longer in active use.

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) MTP - the Mail Transfer Protocol. Used by Laurel (the Alto-based message system) to ship messages to and from mailbox servers.

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, server Altos 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,

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 Maxc1 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. Once you begin to write programs (and run out of local disk space), you will soon discover the importance of these facilities in our daily computing.

Alto software

The first high-level programming language used on the Alto was BCPL, and an overwhelming majority of Alto cycles is still consumed by BCPL programs. Other languages exist however: 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 version of LISP, and others. Mesa and Smalltalk are the most widely used.

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-news.press).

The early development of the Alto programming environment was influenced considerably by existing software for the Nova. We still see this influence in a few places, e.g., certain file-name extensions. However, the desire to communicate with other Altos and with Maxc has had a stronger effect on software written in recent years, and the importance of the Nova is now mostly historical.

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. It is also 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 conjuction with subsystem invocation. You should also read about control-X and control-U in the Executive section of {10}.

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, and is in different places on Alto I's and II's. On Alto I keyboards, it is at the extreme *lower* right, next to the shift key; on Alto II's, 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. To get back to where you were, use the Executive command "Resume". 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. And just to keep you on your toes, only the left-hand Shift key works here! 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.

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. 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.

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. Forget about this case for now. 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. In addition, many Alto programs don't support version numbers (notably those written in Mesa, for low-level implementation reasons). Unfortunately, you don't find out which ones they are until it is too late.

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 and Markup maintain 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 possibly the last one) is not full. Fortunately, it is hard to do so.

Wrinkle #4: In the directory, all file names have a terminating ".". Within the layers of software that access the directory, some define interfaces that expect the terminating period, others supply it automatically. Not all subsystems mask these internal differences effectively; thus, various programs have different ideas about what "ArbitraryFileName" and "ArbitraryFileName." mean. Ah, the wonders of information hiding and abstract specifications...

Wrinkle #5: 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 obscure reasons, 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 #6: 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.

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 man-years 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 - they rarely see one with more than a few hundred available. You will quickly discover that many programmers spend a significant fraction of their day switching disk packs or running FTP. Some do both. 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. Still, if you want to edit, you've pretty much got to have it. 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). Even though Ears is effectively dead, many Non-Programmer disks still have a collection of .EP fonts. These are used only to format documents for Ears, and by deleting them you recover 450 pages. 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. DDS requires 275 pages, and you rarely need its generality; you'll probably prefer to use Neptune or just 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 *or* DDS.run DDS.vmem
Laurel.image RunMesa.run (and various .dl files)
various .al font files

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 /N and /U 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 quickly fade away when these machines arrive on the scene. Mesa today may be likened to a size 12 foot in a size 11 shoe.

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"), and work continues to develop a truly portable Smalltalk machine.

There have been many different releases of Smalltalk, but there have been two principal designs:

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.

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). 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.

The new kid on the block is an Alto-based message system named Laurel. {17} Regular Maxc users still rely primarily on MSG/SNDMSG, but Alto users now have no need to Chat to Maxc periodically to inspect their mail boxes. They do need Maxc accounts, however, to hold their mailboxes.

However, because Laurel stores mail locally on the Alto disk and requires a moderate amount of disk space (about 450 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.

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 new-comer. 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, Versatec, . . . To make matters worse, each of the *instances* of each of these printers has a name as well: Clover, Menlo, Palo, 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.

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 can operate a Dover and a number of other printers, 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, 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
Alto	Text	Press file/printer	Empress, Bravo
	Bravo	Press file/printer	Bravo
	Press	Press printer	Empress
	Draw	Press file	Draw
	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 (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. Perhaps the most notable non-Alto on our network is MAXC, which is a home-grown microprogrammed processor masquerading as a PDP-10. Actually, we have two of them, Maxc1 and Maxc2, implemented rather differently but software compatible. Both run Tenex. Both are 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 for mail either.

Maxc provides three essential services. First, at present, it serves as our only mailbox server. Second, it acts as a file server, archive facility, and software distribution center. Third, it provides the only computationally feasible place to run LISP. Few programming languages other than LISP are exercised on Maxc.

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, but Maxc is the best place to start. Browse through the glossary at the end of this document to get a rough idea of what's around.

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", 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), 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, e.g.,

```
<Packages>*>*.press
```

you will find all sorts of things you might not otherwise locate.

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 software systems in clever, novel, and therefore unanticipated ways ... with predictable consequences.

"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 acquired the value zero when it should had 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. The name comes from the concept of "dealer's choice" - the dealer sets the ground rules and topic(s) for discussion. When someone says he will "give a Dealer on X", he means that he 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.

Once a month, normally on the first Wednesday, Dealer is extended and becomes CSL Luncheon. On these occasions, we have a tasty buffet lunch amidst bean-bag splendor, and Mother Xerox picks 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 SSL Lore

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

The work of the System Science Laboratory spans a wide range of computer-related research, organized into several projects including: the Learning Research Group (LRG), the Office Research Group (ORG), LSI Area, Applied Information Processing Psychology (AIP), a Graphics project, and the Engineering and Software Group.

A Lab-wide meeting is often scheduled on one afternoon each month or two, usually accompanied by something delicious to eat (plan ahead -- find out if a particularly extravagant pastry will be served, and you can skip dessert at lunch!).

A Glossary of Alto Terms, Subsystems, Directories, and Files

(and acronyms, protocols, and other trivia)

Note: For users in Palo Alto, **Maxc**1 is the file server on which most of the directories described below reside. Other locations have equivalent directories on their local file servers (**IFS**s).

<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 **<AltoSource>** and **<AltoDocs>**, respectively.

<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 (ASCII). See **<AltoSource>** and **<Alto>** for corresponding source and object files.

<AltoFonts> A directory on which screen fonts for the Alto are stored (extension .AL).

<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.

ASD Acronym for A_dvanced S_ystems D_epartment, a part of **XBS**.

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.

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

BITBLT (prounouced "bit-blit"). A complex Alto instruction used for moving and possibly modifying a rectangular bitmap. The "BLT" part is an acronym for BLock T_ransfer.

bitmap Generally refers to a representation of a graphical entity as a sequence of bits directly representing points. The Alto display hardware and microcode process what is essentially a bitmap of the image to be displayed.

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.

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.

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 to a (nominally responsible) person.

Cedar A large project in CSL revolving around an experimental programming environment for essentially all of CSL's applications. The system is based on the **Mesa** programming language and many ideas from the **InterLISP** programming environment.

Chat.run 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 Maxc1, Maxc2, and **IFS** servers.

Clover A **Dover** used primarily by **CSL**.

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

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

CSL Acronym for C_omputer S_cience L_aboratory, a part of **PARC**.

Daisy A **Dover** used primarily by **SDD**/Palo Alto.

DDS.run Acronym for D_escriptive D_irectory S_ystem. 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.

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.

DMT.boot Acronym for D_ynamic M_emory T_ester. 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.

Dorado A high-performance computer being designed and built by CSL and intended to execute **Mesa**, **Lisp**, and **Smalltalk** programs very rapidly.

Dover A laser-scan printer built on the Xerox 7000 xerographic engine and connected to an Alto by means of a **Orbit** interface. Successor to **EARS.**

Draw.run An Alto **subsystem** that permits interactive construction of pictures composed of lines, curves, and text.

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 D__o W_hat I_ M_ean. Also, a facility intended to make LISP do what you mean, not what you say.

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

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 (incorrectly) applied to the entire network accessible through the cooperation of **Gateway**s. Every Ethernet within a (connected) network has a unique identifying number.

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
boot	program invokable by boot button
br	BCPL object program module
bravo	text file containing Bravo formatting information
cm	Executive command file
dl	Laurel distribution list
dm	dump file (i.e., several logical files stored as one)
error(s)	Swat error message file
image	executable (Mesa) program
log	history of certain program actions
mail	Laurel mail file
mail-dmsTOC	Laurel table-of-contents file
run	executable (BCPL) program
mesa	Mesa source program module
press	Press file
st	Smalltalk source program text
symbols	Mesa symbol table (for debugging)
syms	BCPL symbol table (for debugging)
ts, typescript	typescript file (log of interaction)

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

<Fonts> A **MAXC** directory containing fonts used for printing (in **Press** files), as well as **Fonts.widths**.

Fonts.widths A file containing character-width information for a large number of fonts. Used by many programs on the Alto that manipulate multiple fonts.

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

FTP.run Acronym for F_ile T_ransfer P_rotocol (or P_rogram). 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 network.

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

IFS Acronym for I_nterim F_ile S_ystem. An Alto-based file server. Several distinct IFS servers exist on various **Ethernet**s, including **Ivy**, **Iris**, and **Isis**.

install A term applied to the Alto Operating System and a number of **subsystem**s (notably **Bravo**), referring to a procedure whereby certain configuration options are established.

InterLISP An interactive version of Lisp with a large library of facilities and a 15-pound reference manual.

Iris An IFS server in **SDD**/Palo Alto.

Ivy An IFS server in **PARC**.

Juniper A **CSL** project and system exploring issues in the design and construction of a distributed file system.

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

KBA Acronym for K_nowledge-B_ased A_ssistance. Refers to an on-going project in **CSL**.

KRL Acronym for K_nowledge R_epresentation L_anguage. Refers to an on-going 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.image An Alto-based, display-oriented message system.

level i system (for i ∈ [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

LRG Acronym for L earning R esearch G roup, a part of **SSL**.

Markup.run An Alto **subsystem** for editing **Press** files.

MAXC Acronym for M ulti-A ccess X erox C omputer (pronounced "Max"). The generic name given to local implementations of a computer functionally similar to the DEC PDP-10. When used in reference to a specific machine, MAXC1 is usually intended.

Menlo A **Dover** used primarily by **SSL**.

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 **PARC** and **SDD**.

<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>**.

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

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**, and a number of diagnostics.

network address A pair of numbers <network number, host number> that uniquely identifies any computer on the network.

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 O perating S ystem. 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.

PARC Acronym for P alo A lto R esearch C enter.

Phogg.image A program that computes the **fog index** for a piece of prose text.

Pine See **PLAP**.

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.

PLAP Acronym for P_age L_evel A_ccess P_rotocol. Sometimes called **Pine**.

Poplar An interactive programming language system running on the Alto, an experimental system in the direction of programming by relatively naive users. Useful for text manipulation applications.

Press A file format used to encode documents for printing and editing. Also, a printing server capable of obeying all the specifications present in a Press file.

PressEdit.run A subsystem that recombines **Press** files on a page-by-page basis. PressEdit can also convert an .ears file to Press format.

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.

<Printing> A **MAXC** directory containing printing and graphics programs.

<PrintingDocs> A **MAXC** directory containing documentation related to printing and graphics facilities such as **Press** files and fonts.

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

800	typewriter-based, word-processing terminal
850	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
6500	20 sec/page copier, color copying
7000	1 sec/page copier
9200	offset-quality, .5 sec/page copier
9700	offset-quality, .5 sec/page, laser-scan printer

PUP Acronym for P_ARC U_niversal P_acket. The structure used to transmit blocks of information (packets) on the **Ethernet**. Also, one such unit of information.

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**.

Scavenger.boot A program available through the **NetExec** that checks for damaged file structures on an Alto disk and tries to repair them.

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 S_ystem D_evelopment D_ivision, a part of **XBS**.

SIL.run Acronym for S_imple IL_lustrator. An illustrator program used for logic design and drawing in general.

<Secretary> A **MAXC** directory containing standard distribution lists for use with **Laurel** and **SNDMSG**.

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

Smalltalk An integrated programming system for the Alto developed by **LRG**.

Spruce A program that recognizes certain portions of a **Press** file (primarily text and **bitmap**s), converts them to a form acceptable by an **Orbit** interface, and prints them.

SSL Acronym for S_ystem S_cience L_aboratory, a part of **PARC**.

<SubSys> A **MAXC** directory containing standard **TENEX subsystem**s.

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.

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**.

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

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

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 **install**ation time (e.g., **Bravo**).

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

XBS Acronym for X_erox B_usiness S_ystems.

XGP Acronym for X_erox G_raphics P_rinter. An obsolete, low-resolution continuous paper, xerographic printer.

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>AltoNetwork.press. Contains a one-page picture of the entire network configuration.

{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} [Maxc]<DMS>Laurel.press. Documentation for the Alto-based, electronic mail system.

{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. and Mitchell, J. G. **Mesa Language Manual.** Xerox PARC Internal Report, October, 1977. 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 [Maxc]<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.