

**Stanford Department of Computer Science
System Documentation**

September 1980

**WELCOME TO ALTO LAND
Stanford ALTO User's Manual**

by

**Stanford Department of Computer Science
and
Xerox Corporation**

CONTENTS

- (1) Preface
- (2) Welcome to the Stanford ALTO User's Manual
- (3) A Field Guide to Alto-land
or Exploring the Ethernet with Mouse and Keyboard
- (4) Alto Non-programmer's Guide
- (5) Bravo Manual
- (6) Alto PUP File Transfer Program
- (7) Laurel Manual
- (8) Alto Subsystems Catalog

**DEPARTMENT OF COMPUTER SCIENCE
Stanford University**



1

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: preface.bravo

Creation date: September 20, 1980 11:25 AM

For: fatis

2 total sheets = 1 page, 1 copy.

PREFACE

This manual is an introduction to the use of the Alto computers at Stanford. It contains a number of documents which should give you some orientation and acquaint you with the basic facilities available.

The *Stanford User's Manual* contains information specific to Alto use at CSD, and a number of caveats worth noting.

A Field Guide to Altoland is a Xerox document which provides many helpful pointers to further information, as well a good deal of background material and general info about Altos. However, since it was intended for Xerox employees, many details are irrelevant to usage at Stanford. Read the *Stanford User's Manual* first.

The *Alto Non-programmer's Guide* introduces you to the Alto executive program and a number of operating system functions necessary to maneuvering yourself around in Altoland. Skip section 2; it's not applicable here.

The *Bravo Manual* is a fairly complete guide to Bravo, the main Alto editor.

Alto PUP FTP explains the file transfer program for communicating with the *Interim File Server* and other Alto disks.

Laurel Manual documents Laurel, the Alto mail handling facility. Don't overlook the on-line Laurel tutorial, which you can access from the executive by typing:

FTP IFS Ret <Laurel>LaurelNewUser.cmCR
@LaurelNewUserCR

Finally, the *Alto Subsystems Catalog* contains pointers to documentation of the major Alto subsystems. Note, however, that some of the subsystems mentioned are not available at Stanford.

Happy Hunting!

2

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: welcome.bravo

Creation date: September 20, 1980 10:17 AM

For: falis

4 total sheets = 3 pages, 1 copy.

Welcome to ALTO land!

Stanford ALTO user's manual

The Stanford University Computer Science Department has 16 ALTO computers available for use, plus an *Interim File Server* (IFS, nicknamed Lassen), and a high quality printer (DOVER, nicknamed Tahoe). This document explains how to get started, where to go for information, and special features of the Stanford Network. There is no current local ALTO expert. Copies of the *Alto User's Handbook* and *Mesa Language Manual* are available from Connie Stanley in the publications office, room 206 for a \$5 deposit.

Before using an Alto it is recommended that new users read those parts of the *Alto User's Handbook* which concern them. Note, however, that the *User's Handbook* assumes you are working for Xerox, and the situation at Stanford is slightly different. Thus, ignore section 2 of the *Non-programmer's Guide*, and all references to *Maxc Field Guide to Altoland* also contains much useful information, but it also is oriented to Xerox employees, so many details do not apply here.

At Xerox, each user has his/her own Alto and several disk packs. At Stanford, the Altos are publicly available, and each has two general-purpose disks. It is the user's responsibility to save and restore the files needed during a specific session. Any user is welcome to use any Alto which displays a Stanford Symbol (a little tree wandering randomly around the screen). If the yellow *ready* lights are not on for both disk drives, switch them to the *run* position and wait for the *ready* lights to come on. Boot the Alto by pressing the button on the back of the keyboard between the cables. When you are done type QuitCR to the executive, and turn the disks back to *load* when the Stanford symbol reappears. If another user will be using the Alto soon, don't bother turning the disks off. Just have the next user type in the LoginCR command.

The directory <AltoDocs> contains documentation that users can print. For example, this document is stored under <AltoDocs>Welcome.Press. To print it, use Chat to log on to Lassen and type Print <AltoDocs>Welcome.PressCR, and then Host DoverCR. Please **do not** print manuals longer than 50 pages on the Dover without asking around for someone else's copy to borrow.

The <Forms> directory contains several useful forms. The Stanford CSD letterhead is stored under <Forms>Form.Letter. <Forms>Form.PersonalLetter should be used for non-official letters. TitleBlock.Sil is used to title logic diagrams done in CSD. Schedule.Draw is a Draw format outline for a class or office hours schedule.

Accounts

Our grant from Xerox states that the ALTOs should be used for teaching and research applications. Accounts are needed to bootstrap and to use the Stanford disk, and to obtain permanent storage on the IFS. If you want an Alto account, fill out "new accounts" form and return it to Susan Hill.

After booting a disk, type your name, return and your password. The password is checked for validity as an IFS account and remembered by the operating system. When you are finished with a session, another user can use the Alto by typing a LoginCR command.

Mail

The Laurel mail system is used to announce new changes and additions to the system. When you log in, you will be notified if you have any unread mail. New messages are queued on the Lassen directory <Mail>Box> and are fetched automatically by Laurel. The problem is that Laurel expects to keep a working mail file on the local disks. We have solved this problem by writing a command file that FTP's the working file to and from Lassen. To use it type "@Mail" instead of "Laurel". After retrieving your Active.Mail file, the command file invokes Laurel with an automatic check for new mail. After you quit from Laurel your Active.Mail file is returned to the IFS. Note that if the next person to use your Alto just runs Laurel instead of using the command file, he will get your mail file.

@Mail has been changed to delete Active.mail files and fetch distribution lists from the IFS. The distribution list bboard contains every known user, so an effect similar to the SCORE bboard can be achieved by sending to bboard†.

Other System News

You can now print press files from Lassen directly without first retrieving them with FTP. Use the Print command from FTP Telnet or Chat. Due to an IFS bug you must specify the printing host Tahoe (or Dover) each time the Print command is used. The *FTP Reference Manual* to be found in *Alto User's Handbook* may be outdated. Check <AltoDocs>Ftp.Tty on-line for more current info.

Several command (.cm, .do) files have been added that make certain common commands easier to do. @Clean will clean-up the local disk by deleting non-permanent files. "Do Get <filename>" will do an FTP retrieve of <filename> from Lassen. "Do Put <filename>" will do an FTP store to the IFS. A few new fonts have been installed on the Dover. One of these is Stan200, default font #2 in Bravo. It contains the Stanford seal as character S, and the letters "c", "i" and "s" for a logo for the Center for Integrated Systems.

Complaints, comments, suggestions, improvements, etc. should be sent to your local guru, if you can find one. This manual was written by Bill Nowicki, May 1980 and revised by Ed Falis, September 1980.

Network Names

All the EtherNet hosts (including Altos) have been given names. The list is stored on <System>Pup-Network.txt. Names correspond to mountains, valleys, lakes, and wilderness areas within California. The name of your Alto should appear in one of the operating system's header lines. Locations are subject to occasional changes.

Host Computer Names

Yolo	MJH 460	(ALTO/Terminal room)
Inyo	MJH 460	
Yuba	MJH 460	
Mono	MJH 460	
Diego	MJH 225	(HPP terminal room)
Mojave	MJH 408	
Yosemite	MJH 408	
Marin	MJH 450	(Office Suite)
Monterey	MJH 450	
Madera	MJH 433	(SUMEX)
Trinity	MJH 433	(Music room, for debugging VAX interface)
Toro	MJH 416	(Student jungle)
Napa	MJH 416	
Almanor	MJH 328	(NA terminal room)
Palomar	MJH 020	(Machine room, for debugging SAIL interface)
Lassen, Stanford-IFS, IFS	MJH 020	(File Server, in machine room)
Tahoe, Stanford-Dover, Dover	MJH 221	(Dover printer, in Xerox room)
Shasta, VAX, VAX1, VLSI1	MJH 433	(Center for Integrated Systems computer room)
VAX2, VLSI2	MJH 433	
VAX3, ISL-VAX	Durand	(Information Systems Laboratory)
Sail, SU-AI	MJH 020	(KL-10 in machine room)
Score	MJH 020	(DEC 2060 in machine room)
HF, 11/45	MJH 029	(Robotics group PDP-11/45)

3

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: fieldguide.press

Creation date: 20-Sep-79 11:00:20 PDT

For: falis

32 total sheets = 31 pages, 1 copy.

Problems encountered:

Font LOGO18 substituted for font LOGO24.

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

© 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. A lot 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, 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.

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 downplay 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 (Parc Universal Packets) {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 Maxc and two NFS 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 conjunction 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 Hardecopy 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.

(Genealogists 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

<i>Running on</i>	<i>Input file format</i>	<i>Output desired</i>	<i>Program to use</i>
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, **Max1** is the file server on which most of the directories described below reside. Other locations have equivalent directories on their local file servers (**IFSs**).

<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 <u>Advanced Systems Department</u> , 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	(pronounced "bit-blit"). A complex Alto instruction used for moving and possibly modifying a rectangular bitmap. The "BLT" part is an acronym for <u>B</u> lock <u>T</u> 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 <i>report</i> 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 Computer Science Laboratory, a part of PARC.
Daisy	A Dover used primarily by SDD/Palo Alto.
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.
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 Dynamic 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.
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 <i>Swatee</i> for subsequent inspection by a debugger.																																												
DWIM	Acronym for <u>Do What I Mean</u> . 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 Gateways. 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:																																												
	<table border="0"> <tr> <td>~</td><td>an Executive command (not really an extension)</td></tr> <tr> <td>al</td><td>Alto screen font</td></tr> <tr> <td>bcd</td><td>Mesa object program module</td></tr> <tr> <td>bcpl</td><td>BCPL source program module</td></tr> <tr> <td>boot</td><td>program invokable by boot button</td></tr> <tr> <td>br</td><td>BCPL object program module</td></tr> <tr> <td>bravo</td><td>text file containing Bravo formatting information</td></tr> <tr> <td>cm</td><td>Executive command file</td></tr> <tr> <td>dl</td><td>Laurel distribution list</td></tr> <tr> <td>dm</td><td>dump file (i.e., several logical files stored as one)</td></tr> <tr> <td>error(s)</td><td>Swat error message file</td></tr> <tr> <td>image</td><td>executable (Mesa) program</td></tr> <tr> <td>log</td><td>history of certain program actions</td></tr> <tr> <td>mail</td><td>Laurel mail file</td></tr> <tr> <td>mail-dmsTOC</td><td>Laurel table-of-contents file</td></tr> <tr> <td>run</td><td>executable (BCPL) program</td></tr> <tr> <td>mesa</td><td>Mesa source program module</td></tr> <tr> <td>press</td><td>Press file</td></tr> <tr> <td>st</td><td>Smalltalk source program text</td></tr> <tr> <td>symbols</td><td>Mesa symbol table (for debugging)</td></tr> <tr> <td>syms</td><td>BCPL symbol table (for debugging)</td></tr> <tr> <td>ts, typescript</td><td>typescript file (log of interaction)</td></tr> </table>	~	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)
~	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 <u>F</u> ile <u>T</u> ransfer <u>P</u> rotocol (or <u>P</u> 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 Ethernets. Gateways may also perform certain server functions, such as name lookup.
IFS	Acronym for <u>I</u> nterim <u>F</u> ile <u>S</u> ystem. An Alto-based file server. Several distinct IFS servers exist on various Ethernets, including Ivy , Iris , and Isis .
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.
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 <u>K</u> nowledge- <u>B</u> ased <u>A</u> sistance. Refers to an on-going project in CSL .
KRL	Acronym for <u>K</u> nowledge <u>R</u> epresentation <u>L</u> 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 $i \in [1..3]$). A terminology for classifying (software) systems according to their intended user community:

- 1 implementors only
- 2 implementors and friendly users
- 3 naive users

LRG	Acronym for <u>L</u> earning <u>R</u> esearch <u>G</u> roup, a part of SSL.
Markup.run	An Alto subsystem for editing <u>P</u> ress files.
MAXC	Acronym for <u>M</u> ulti- <u>AXC</u> 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 <u>D</u> over 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 <u>n</u> etwork <u>a</u> ddress. 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 <u>O</u> perating <u>S</u> 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 <u>P</u> alo <u>A</u> lto <u>R</u> esearch <u>C</u> 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 <u>P</u> age <u>L</u> evel <u>A</u> ccess <u>P</u> 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:																				
	<table><tbody><tr><td>800</td><td>typewriter-based, word-processing terminal</td></tr><tr><td>850</td><td>display-based, word-processing terminal</td></tr><tr><td>2600</td><td>desktop copier</td></tr><tr><td>3100</td><td>3 sec/page copier, good solid black-area development</td></tr><tr><td>4500</td><td>1 sec/page copier, 2-sided copying</td></tr><tr><td>5400</td><td>1 sec/page copier, good resolution</td></tr><tr><td>6500</td><td>20 sec/page copier, color copying</td></tr><tr><td>7000</td><td>1 sec/page copier</td></tr><tr><td>9200</td><td>offset-quality, .5 sec/page copier</td></tr><tr><td>9700</td><td>offset-quality, .5 sec/page, laser-scan printer</td></tr></tbody></table>	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
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 PARC Universal Packet. 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 <u>S</u> ystem <u>D</u> evelopment <u>D</u> ivision, a part of XBS.
SIL.run	Acronym for <u>S</u> imple <u>I</u> llustrator. 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 bitmaps), converts them to a form acceptable by an Orbit interface, and prints them.
SSL	Acronym for <u>S</u> ystem <u>S</u> cience <u>L</u>
<SubSys>	A MAXC directory containing standard TENEX subsystems.
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 installation 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 <u>Xerox</u> <u>B</u> usiness <u>S</u> ystems.
XGP	Acronym for <u>Xerox</u> <u>G</u> raphics <u>P</u> 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>FTPsing.spec.press. A functional specification of the file transfer protocol, independent of implementation in any particular language or system.
- {5} [Maxc]\Pup>EFTPsing.spec.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, FIP, 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]<DM>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.

4

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: nonprogguide.press

Creation date: 14-Dec-79 12:06:01 PST

For: falis

31 total sheets = 30 pages, 1 copy.

Problems encountered:

Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.

Alto Non-programmer's Guide

Table of Contents

1. Introduction	2
2. Getting started	3
3. The Executive and the NetExec	5
3.1 Correcting typing errors	5
3.2 Starting a program	5
3.3 Aborting	6
3.4 The NetExec	6
4. Files	7
4.1 Naming conventions	7
4.2 File name patterns	7
5. Recovering from disasters	9
5.1 Reporting problems	10
6. File servers	11
6.1 Logging in	11
6.2 About files on IFS and Maxc	12
6.3 Transferring files	12
6.4 Listing and deleting files	14
6.5 Transferring files to and from another Alto	14
6.6 Access via Chat	14
6.7 Server Executive commands	16
6.8 About Maxc	17
7. Printing	19
7.1 Programs and file formats	19
7.2 Printing servers	19
7.3 Fonts	20
7.4 Printing from your Alto	21
7.5 Printing from Maxc	22
7.6 PressEdit	22
8. Other things	24
8.1 Copy and Rename	24
8.2 Command files	25
8.3 Dump files	25
8.4 Neptune and DDS	26
8.5 Illustrators	26
8.6 CopyDisk	27
8.7 Version numbers	27
9. Software distribution and documentation	28
9.1 Obtaining new software releases	28
9.2 Documentation	29

1. Introduction

This document is intended to tell you what you need to know to create, edit, and print text and pictures on the Alto. It doesn't assume that you know anything about Altos, Maxc, IFS, or any of the other facilities available to you.

You will find that things are a lot clearer if you try to *learn by doing*. This is especially true when you are learning to use any of the services which use the display. Try out the things described here as you read.

Material in small type, like this paragraph, deals with fine points which can be skipped on first reading (and perhaps on subsequent readings as well).

Much of the documentation in this Guide is intentionally incomplete. More comprehensive information about almost all of the programs and services described here may be found in various on-line documents. Section 9 contains a summary of these and instructions for obtaining your own copies of any that you need.

2. Getting started

To do anything with an Alto, you must have a disk pack. This is a circular, yellow or white object about 15 inches in diameter and 2 inches high. Your secretary can tell you how to obtain a new one from the stock kept by your organization.

INITIALIZE YOUR DISK

The next step is to get the disk *initialized* with copies of all the programs you will need to use. Here is how to do this:

Obtain the disk pack labeled BASIC NON-PROGRAMMER'S DISK. Find an Alto that has two disk drives, each with four square lights, a white switch and a slanted plastic window. Load the BASIC NON-PROGRAMMER'S DISK into the drive labeled 0. You do this as follows:

The drive should have the white switch in the LOAD position, and the white LOAD light should be lit. Open the door by pulling down on the handle. Put in the disk by holding it flat, with the label facing you, and pushing it gently into the drive until it stops. Then gently close the door and push the white switch to RUN. The white LOAD light will go out, and after about a minute the yellow RUN light will go on. The disk is now loaded and ready to go. If anything else happens, you need help.

On many double-disk Altos, the two disk drives are not labelled. The drive mounted inside the same cabinet as the Alto is drive 0, and the one sitting on top of the cabinet is drive 1. Also, some Altos have a switch on the back of the keyboard housing labelled NORMAL/ALTERNATE. Be sure that this switch is in the NORMAL position.

Now start the Alto. This is done by pushing the small button on the back side of the keyboard, near the thick black cable. Pushing this button is called *booting* the Alto. It resets the machine completely, and starts it up working on the disk you have just loaded. After you boot the machine, it will tell you at the top of the screen what it thinks the state of its world is, and then it will print a ">" about halfway down the screen. When the screen looks like that, anything you type will be read by the Executive, whose basic job is to start up the service you want to run. There is a section on the Executive later in this document. For now, you will find everything you need to know right here.

You are going to use a program called CopyDisk, which copies everything on the main disk (which you just loaded) onto another disk which you will load into the disk drive labeled 1. This copying erases anything which is already on the disk in disk drive 1, so you should be very careful not to copy onto a disk which has anything you want to keep. Load your new disk into the disk drive labeled 1, doing just what you did to load the BASIC NON-PROGRAMMER'S DISK into drive 0.

The CopyDisk program is not present on the BASIC NON-PROGRAMMER'S DISK, but it is available through a service called the NetExec, which can load a small number of commonly-used programs from the Ethernet. To start the NetExec, type

>NetExecCR

The CR stands for the carriage RETURN key on the keyboard. In this and later examples, what you type is underlined in the example, and what the Alto types is not. On the screen, of course, there won't be any underlining. It doesn't matter whether you capitalize letters or not; the capitalization in this manual is chosen to make reading easier.

Within a few seconds, the NetExec will start up. The screen will look much the same as it did while the Executive was running, but with "Net Executive" in the upper left corner. Now type

>CopyDiskCR

After a few more seconds, CopyDisk will start up, identify itself, and display a prompt of "*". You

should now go through the following dialogue:

*Copy from: DP0CR *the digit zero, not the letter O*
Copy to: DP1CR
 Copying onto DP1 will destroy its old contents.
 Are you sure this is what you want to do? [Confirm] Yes
 Are you still sure? [Confirm] Yes

Now CopyDisk will copy the contents of the BASIC NON-PROGRAMMER'S DISK (DP0, "the Disk Pack in drive 0") onto your new disk pack (DP1). This takes about two minutes. While it is running, it records its progress by moving the cursor from the top of the screen to the bottom; this happens twice: once while the disk is being copied and again while the copy is being checked for correctness. When it is done, if all went well it will display the message "Done. DP0 and DP1 are now identical." followed by the "*" prompt. Now type

*QuitCR

to exit CopyDisk and return control to the Executive. If something goes wrong, the message "Copy complete, but do *not* trust DP1" will appear. This means that there is something wrong either with the Alto or with one of the disk packs. Consult your local support staff.

Now you can take both disks out of the machine. Before you do, you should tell the Executive that you are finished, by typing

>QuitCR

You will see that after a couple of seconds the screen goes blank and starts to display a white square that jumps around. This is an indication that the *memory test* program is running properly; an Alto should always be left in this state when it is not being used.

Now take out both disks, by pushing the white switch on each drive to LOAD. The yellow READY light should go out, and about 25 seconds later the white LOAD light should go on. Now you can open the door (against a *slight* resistance) and remove the disk. Put the BASIC NON-PROGRAMMER'S DISK back where you found it

If you cannot find an Alto with two disk drives, you can do a CopyDisk from one standard (single-drive) Alto to another; the procedure for doing this is described in section 8.6. Since it is a little more complicated than the method just given, a novice should use it only as a last resort.

LABEL YOUR DISK

Before doing anything else, put a label on the new disk with your name, and any other identifying information you like. This is best done by preparing a paper label that can be slipped underneath the plastic insert on the front edge of the disk pack. Now you can take the new disk to any Alto, load it in, boot the machine by pushing the button on the back of the keyboard, and start working.

NAME YOUR DISK

When you do this, if you look at the information displayed at the top of the screen just after you do the boot, you will see that it says

---- OS Version x/x ---- Alto #xxx ---- NoName ---- Basic Non-programmer's Disk ----

This is because your new disk is an exact copy of the BASIC NON-PROGRAMMER'S DISK, which has no owner, and owner and disk name information got copied along with everything else. To give the disk your own name as owner, you should type

>InstallCR

to the Executive. It will ask you whether you want the "long installation dialogue"; answer No. When it asks you for your name, type in your Maxc or IFS account name (usually just your own

last name), followed by a CR. When it asks you for a disk name, choose a suitable one and type that in, again followed by a CR. Next it will ask you whether you want to give your disk a password. If you do this, the Alto will ask you for the password every time you boot it, and won't let you do anything until you provide it correctly. This provides a modest level of security for the information on your disk. If you do give your disk a password, it is best to use your Maxc or IFS password, since the Alto will then know it and use it automatically whenever you communicate with Maxc or IFS. *Don't forget the password, since there is no simple way to find out what it is, and you will need an expert to get access to anything on your disk.*

There will be a pause for a few seconds, and then the Executive will come back. If you assigned a password to your disk, you will be asked for it first. Now your name is installed on the disk, and the system will display it near the top of the screen whenever the Executive is in control, and will put it on the cover page of anything you print.

After you initialize a disk, you have to edit your *user profile*, discussed in section 2.4 of the Bravo manual, and your *Laurel profile*, described in section 3.6 of the Laurel manual. If you are reading this manual for the first time, you will be told how to do these things at the appropriate points. This is mentioned here so that you will remember it next time you initialize a disk.

3. The Executive and the NetExec

The Executive is the program to which you are typing right after a boot, and whenever any other program finishes its job. It has a large display area in the middle where your typing and the Executive's responses appear. Above this the Executive displays a digital clock and some other useful status information: the versions of the Executive and the operating system, the owner name and disk name installed on the disk, the Ethernet address of the Alto you are using, and the number of free pages on the disk. Whenever you call another program, the Executive's display is erased and replaced by that of the program that you called.

3.1 Correcting typing errors

When you are typing at the Executive and you make a mistake, there are a few special keys you can type to correct the mistake. The BS (backspace) key erases the last character you typed. Holding down the CTRL key and typing W erases the last word you typed. The act of holding down CTRL and typing W is called control-W and is denoted by W^c , and similarly for other control characters. The DEL key cancels the command you were typing completely; it prints "XXX", and then starts a new line with a fresh ">" character. Nearly all programs accept A^c (control-A) as a substitute for BS.

3.2 Starting a program

As we said before, the Executive is for starting up other programs which do the work you want done. To start a program called Alpha, you just type

>AlphaCR

It doesn't matter whether you type in capitals, lower case, or a mixture of the two. If the program needs some other information about what to do, you type that after the name of the program. For example, there is an Executive command to type out a document on the screen. Suppose you want to type out the document called "Notes". You just say

>Type NotesCR

The Executive won't ever do anything until you type the final CR; if you change your mind, just type DEL to cancel the command any time before you type the CR.

Certain operations, such as Type, are performed entirely by the Executive itself, whereas most others are performed by separate programs (also called *subsystems*) kept on your disk or obtained from the Ethernet. Ignore this distinction for now.

3.3 Aborting

You can usually stop what is going on and get back to the Executive by holding down the left-hand SHIFT key and striking the SWAT key, which is a blank key in the lower right corner of the keyboard on Alto-Is, in the upper right corner on Alto-IIs. If this doesn't work, you can push the boot button.

If you push the SWAT key while holding down both CTRL and SHIFT, you will find yourself talking to a service called Swat which is of no interest to non-programmers. Usually no harm is done if this happens; you can get back to what you were doing before by typing PC (control-P; hold down the CTRL key and type P).

3.4 The NetExec

The NetExec is a program much like the Executive in that its main purpose is to start up other programs for you. Unlike the Executive, it loads programs from a *boot server* available via the Ethernet rather than from your own disk. The NetExec makes available certain programs (such as hardware diagnostics) that are used infrequently and that most users won't wish to keep on their own disks. Also available are several programs useful for recovering from various sorts of disasters that may make it impossible for you to invoke the normal Executive.

The NetExec may be started in either of two ways. If the Executive is already running, you may simply type

>NetExecCR

A fuzzy cursor will appear in the center of the screen for a few seconds, and then the NetExec will start up. If the Executive is not running, you can invoke the NetExec directly by holding down the BS key and the ' (quote) key and then pushing and releasing the boot button. Keep the keys pressed down until you see a fuzzy cursor in the center of the screen; this can take up to 5 seconds.

The NetExec's display looks much like the Executive's, but the herald contains the words "Net Executive". The type-in conventions are identical to the Executive's. To start up a program from the NetExec, simply type the name of that program followed by CR.

Any time this manual instructs you to "use the NetExec to invoke *p*", where *p* is the name of some program, you should follow the above procedure. An example of this was given in the instructions for using CopyDisk to initialize your disk (section 2).

The Executive also has a few commands for invoking programs directly from the Ethernet, without your first having to start up the NetExec. At the present time, these programs are Chat, FTP, and Scavenger. More precisely, these commands will obtain the correspondingly-named programs from your disk if they are present and from the Ethernet otherwise.

4. Files

The Alto stores on your disk all of the material you are working on (text and pictures), as well as most of the programs which provide the various services described here. The named unit of storage on the disk is called a *file*. Each different document you handle will be stored on its own file. The facilities for identifying files are not ideal, but you will get used to them after a while. Better facilities are the subject of current research.

A file is identified by its *name*, which is a string of letters (upper and lower case can be used interchangeably), digits, and any of the punctuation characters + - . ! \$. A file name can have two parts, which are called the *main name* and the *extension*; they are separated by a period. For example, "Alto.Manual" is a file name, with main name "Alto" and extension "Manual". File names cannot have blanks in them, or any punctuation characters except the ones just mentioned. A file name must not have more than 39 characters; most people don't notice this restriction.

4.1 Naming conventions

It is important to name your files in some systematic way, using the extension to tell what kind of file it is, and the main name to identify it. For instance, useful extensions might be Memo, Letter, Note, Figure, Calendar. If you are a secretary keeping material for several people on one disk, you can stick the person's initials in front of the extension, e.g. BWL.memo, JGMmemo etc. If you don't have anything specific in mind, it is customary to make the extension the same as the name of the program that creates the file, e.g., Report.bravo for a document that doesn't have any special properties, and is written using Bravo.

Here is a modest list of extensions commonly encountered on Non-programmers' disks:

.al	Alto display-format font file
.boot	A file that can be booted from
.bravo	Bravo-format text file
.cm	Command file for the Executive or other programs
.image	Runnable Mesa program (subsystem)
.press	Press-format file (suitable for printing)
.run	Runnable Bcp! program (subsystem)
~	An Executive command that is executed directly by the Executive (there is no actual file corresponding to this name).

The Alto doesn't care whether you capitalize letters in file names or not (i.e., ALPHA and alpha and aL.pHa refer to the same file), but it is a good idea to use capitalization to make names more readable. This is especially useful when a name consists of more than one word, since blanks are not allowed in file names: e.g., TripReport or MasterList.

4.2 File name patterns

The Executive provides some simple facilities for handling files. First of all, it allows you to name a group of files by using file name *patterns* containing the magic characters "*" and "#". The "*" character stands for any string of characters. For example, the pattern "* .memo" stands for all the files which have the extension "memo", and the pattern "*.BWL*" stands for all the files which have BWL as the first three characters of the extension. The "#" stands for any single character; for instance, "# # # .memo" stands for all the files which have a three character main name and the extension "memo". If you are curious to see what a pattern expands into, you can type Xc immediately after typing it to get it expanded.

If you type a file name or a pattern to the Executive, and then type a TAB, it will give you a list of all the files whose names start with that name. So, for example, typing

>*.BWLTAB

will get you a list of all files which have an extension starting with the characters BWL.

Another useful thing to know is this: if you are in the process of typing a file name to the Executive, and you type ESC, it will add as many characters as it can to complete a file name. If you type "?", it will give you a list of all the files which start with what you have already typed; you can then go on and finish the file name.

Here is a summary of magic characters for getting file names expanded:

- ESC completes the file name if possible; if not, completes as much as it can, and flashes the screen.
- TAB shows you all the file names which match what you have typed since the last blank, and erases what you typed.
- ? like TAB, but doesn't erase anything.
- X^c retypes the command line with all file name patterns replaced by the list of file names they expand to.

There are two more simple commands for dealing with files. To delete a file, or a group of files, type

>Delete *file1 file2 ...CR*

Warning: once you have deleted a file, you cannot get it back. Proceed with caution. If you have enabled version numbers and there is more than one version of a file, the one with the lowest version number gets deleted.

To get the contents of a text file printed on the screen, type

>Type *fileCR*

If the contents won't fit on the display, the Alto will show you as much as will fit, then ask if you want to see more. If you do, just type a space; if you want to stop, type No.

When the Executive is running, it displays two lines of status information near the top of the screen. Included in this information is the amount of space which is left for storing files. This space is measured in *disk pages*; it takes about 5 disk pages to store one page of text. It is prudent to keep at least 150 disk pages available; if your disk has fewer, you should delete some files, perhaps after sending them to a file server (see section 6).

At this point you know enough to use Bravo to begin creating and editing text. Bravo is described in its own manual, which is part of the Alto User's Handbook. You should start reading the Bravo manual, and not try to continue with this guide until you have become familiar with the material in the first two sections of the Bravo manual. The remainder of this guide contains more information about the Alto which you won't need on the first day, but will probably want in the first week.

Because much of our day-to-day communication takes place by means of our Alto-based electronic mail system, you should also start learning to use Laurel. Begin by reading the first two sections of the Laurel manual, which is also part of the Alto User's Handbook.

5. Recovering from disasters

There are various ways in which your Alto disk can become damaged. If this does happen, the procedures described in this section will almost always allow you to recover the disk, or at worst will let you copy files from the sick disk to a healthy one. It is probably a good idea to get some help with this if you are not experienced.

Here are the symptoms of trouble:

You can't boot the disk and get to the Executive.

You are out of disk space, but you think you should have plenty; in other words, some disk space has apparently gotten lost.

You get an error message from some service which says something about disk errors or file errors, and perhaps recommends that you should run the Scavenger.

You hear a funny buzzing noise from the disk for a couple of seconds, after which the service you are using breaks in some way.

It may be that the problem is caused by an incompatibility between the disk drive on which your disk pack was written, and the disk drive on which you are trying to use it. This is a likely cause of your problems only if you have been moving the pack from one machine to another, and if you notice that it works properly on some machines but not on others. If your problem is caused by disk incompatibility, the procedures described below won't do you much good. Instead, you should report the problem to the hardware maintenance staff, so that the offending disk drive can be realigned, and make yourself a new disk pack on a machine known to be in alignment. You can transfer files from the old pack to the new one using the procedure described in section 6.5.

The first step is to run a program called Scavenger. If your disk is healthy enough to let you boot and use the Executive, you can just invoke the Scavenger by typing

>ScavengerCR

If it isn't, you should use the NetExec to invoke Scavenger, as described in section 3.4.

If that doesn't work, hold down just the BS key and press the boot button; this should give you the dancing white square of the memory diagnostic. If it doesn't, either your Alto's Ethernet connection is broken, or the *boot server* that provides Alto programs over the Ethernet is down or unavailable. Either find another Alto without these problems, or load in a disk which is still in good shape and has the Scavenger program on it, invoke the Scavenger, and then unload the good disk and load your sick one.

The Scavenger will ask you whether you want to change disks, and give you a chance to do so if you say Yes. Then it will ask you if it can alter your disk to correct errors; say Yes.

The Scavenger will now work for about a minute. As it runs, it may ask you whether it is OK to correct "read errors". If they are "transient" errors, answer Yes fearlessly; if they are "permanent" errors, it is best to ask for advice from an expert. When the Scavenger is done, it will tell you what it found. If it has succeeded in making your disk healthy, you can go about your work. Delete the files Garbage.\$ and ScavengerLog which the Scavenger leaves around. It is a good idea to go through this scavenging procedure once a month or so, just to keep your disk in good shape.

If things are still in bad shape (i.e., you can't boot and run the Executive), the next step is to boot the NetExec again and type

>NewOSCR

This should get you a fresh copy of the operating system, which will ask you whether you want to Install. You should say Yes, and go through the Install procedure described in section 2. If all goes well, you will then find yourself talking to the Executive and can proceed normally.

If this doesn't work, there is one more step to try. Boot the NetExec again and this time type

>I:TPCR

This should get you the FTP program described in section 6.3; use it to transfer the files <Alto>Executive.run and <Alto>SysFont.al from a file server (Maxc or IFS). Then boot the Scavenger as described above and run it again. If

this fails, you should consult an expert. If no expert is available, you can boot FTP again, and use it to transfer files from your broken disk to a file server or to a clean disk on another Alto (made using the procedure described in section 2).

The Scavenger leaves all the stuff which it wasn't able to put into a recognizable file on a file called Garbage.\$, and it leaves a readable record of everything it did on another file called ScavengerLog (unless it tells you that you have a beautiful disk). There are two kinds of entries in ScavengerLog: names of files removed from the directory or otherwise modified, and names of file pages which were put into Garbage.\$.

5.1 Reporting problems

If your Alto itself is broken, obtain a trouble report form, fill it out, and leave it in the proper place; procedures for doing this depend on your location.

If you have trouble with Bravo, report it using the procedure in section 4.3 of the Bravo manual. If you have trouble with Laurel, see section 4 of the Laurel manual.

For other problems, consult your local expert.

6. File servers

Many uses of the Alto require you to communicate with a shared *file server*, which can store files belonging to a number of different users. A file server's disk typically has hundreds of times the capacity of your own Alto disk. Moving non-current files to a file server is a way to free up space on your own disk for other things, and a file server is a good place to put documents you want other people to be able to get at.

Each organization typically has at least one file server of its own. To make any use of your organization's file server, you must first obtain an account and password; to do this, consult your local support staff. Your account name will usually be your own last name, and your password should be six or more characters long and unpronounceable.

There are presently two types of file server, IFS and Maxc. An IFS (which stands for Interim File System, as improved facilities are under development) is an Alto with some large disks attached to it and dedicated to the one task of being a file server. Many IFSs exist at present. Maxc is a large, general-purpose time-sharing system, only one of whose purposes is to be a file server. There are only two Maxc systems (called Maxc1 and Maxc2), both of which belong to Parc.

If you are at Parc, you should obtain accounts both on Ivy (the name of Parc's IFS) and on Maxc. If you are outside Parc, you will need an account on your organization's own IFS, and you will have to find out its name. At present, many users outside Parc are also assigned accounts on Maxc because the Laurel message system uses Maxc as a central post-office, and you have to have a Maxc account in order to keep a mailbox there. This state of affairs is likely to change in the near future.

While IFS and Maxc are fundamentally different sorts of machines, in their capacities as file servers they are substantially the same. We shall first describe the facilities common to the two types of systems. Later sections will deal with additional facilities unique to one or the other.

6.1 Logging in

Before trying to access a file server from your Alto, you should first tell the Alto your account name and password. If you have given your account name to Install as the owner name for your disk, however, the Alto already knows it, and if you used your file server password as your disk password, it knows that too and you can skip to section 6.2. Otherwise, you can give the necessary information by typing to the Executive:

>LoginCR

You will now be asked for your name and password. Type in each one in turn, ending each with a CR or space. Note that it assumes your file server account name is the same as your disk owner name; if this is the case, you can just type CR to confirm it, and go on to give your password. If it isn't, type DEL, and then give the account name you want to use. Once you have done this Login, your Alto will automatically identify you to file servers whenever necessary. *If you boot your Alto, it will forget this information, and you must Login again.*

Note that Login only records your name and password; it does *not* connect you to a file server. If you don't do a Login, programs that access file servers will automatically ask you for the Login information when they first run, and will record it just as Login does.

If you wish, you can supply a password for your disk when you Install (see section 2). If you do this, you will have to type the password whenever you boot the Alto, but it will be used automatically as your file server password, unless you override it with a Login command. The password is stored on your disk in encrypted form, so it cannot be stolen by someone who paws around on your Alto disk.

6.2 About files on IFS and Maxc

IFS and Maxc file names look very much like Alto file names, but they have two more parts: a *directory* and a *version number*. On IFS, the format is

<directory>name.extension!version

whereas on Maxc the format is

<directory>name.extension;version

Each user who has an account on a file server has his own directory, named by his user name. If you name a file without specifying the directory, you will refer to your own directory. You can reference files in some other directory simply by prefixing the directory name to the file name, as illustrated.

When you put a file onto a file server, if there is already a file with the same name, the new file is added, with a version number one bigger than the old one. When you reference a file, you get the one with the largest version number if you don't specify which one. As you can see, it is almost never necessary for you to specify a version number explicitly.

There is a protection system, not described here, which allows you to control which other users can read or write your files. The usual setting of the protection, and the one you will get automatically if you don't say anything special, allows all Xerox users to read the file, but prevents anyone except the owner from writing it.

On IFS, but not on Maxc, files within a directory may be organized into *sub-directories*. For example, the file named
 <Jones>Memos>ActivityReport.bravo!3

belongs in directory Jones, sub-directory Memos. You can have as many sub-directories as you wish within your own directory. You can even have sub-directories within sub-directories, to as many levels as you wish, subject to an overall limit of 99 characters in each file name.

Most commands that accept a remote file name also permit you to type a *file name pattern*, much as does the Alto Executive (section 4.2). In this case, the command is repeated for all files that match the pattern. If the file server is an IFS, the magic character "*" (but not "#") may appear anywhere in the name, just as on the Alto; if the file server is Maxc, however, "*" must appear in place of an entire field of the file name (directory, main name, extension, or version), and other uses of "*" are not permitted. Thus, "*.memo" and "<Jones>*.*" are legal file name patterns on both Maxc and IFS, but "*!BWL*" is legal only on IFS. Note that to refer to *all versions* of each file you must specify "*" for the version field explicitly (that is, "!*!" on IFS and ";*" on Maxc); otherwise, the pattern will match only one version (usually the highest-numbered).

6.3 Transferring files

You can transfer files between your Alto and a file server using FTP, the File Transfer Program. This program has a fairly elaborate set of features, which are described fully in the FTP Reference Manual found near the end of the Alto User's Handbook. This section tells you enough about FTP to take care of most ordinary needs. After you have become familiar with this material, you might skim over the FTP manual once, just so that you are aware of the other facilities that are available.

After starting FTP, you will see three windows on the screen; from top to bottom, they are the *server* window, the *user* window, and the *Chat* window. Most interactions with FTP involve only the middle window—note the blinking vertical bar there, which shows where you can type. The first step is to type the name of the machine you want to talk to. For example, if your file server is named Ivy, you should just type

*IvyCR

In a second or two you should get back a response like

Ivy (Parc) file server 1.16 9-Apr-78

If the file server is not operating, there will be delay of about a minute before FTP gives up trying to contact it; you can give up sooner by striking the blank key to the right of CR.

Now you can *retrieve* a file from the file server, or *store* a file into it. To retrieve, you type
***Ret_rive remote file Example as local file Example [New file] CR**

As in the Executive, you can just type enough of the command to identify it uniquely, and then a space; unlike the Executive, FTP supplies the rest of the command name automatically. You then type the remote file name, followed by a space. If the file server has a file by that name on your directory, FTP will then suggest a local name for the file followed by "[Old file]" or "[New file]", depending on whether or not the file already exists on your Alto disk. If you like the name, you can just type CR. Otherwise, you can type some other name, as in the following example:

***Ret_rive remote file Example as local file Dummy [New file] CR**

Note that the name originally suggested by FTP will disappear as soon as you start to type the new one. If you decide you don't want to retrieve the file after all, strike DEL to cancel the command.

After you type the terminating CR, FTP will transfer a copy of the file from the file server to your Alto disk. During the transfer, the cursor will flip its two black squares back and forth every time it transfers a block of the file, so you can tell how fast it is progressing from the frequency of flips. When FTP is done, it will tell you how long the file is, followed by "Done" and the "*" prompt. If you gave a *file name pattern* for the remote file, FTP will repeat the above procedure for each file that matches the pattern. You may strike DEL to skip over any files that you don't want to retrieve.

To store a file presently on your Alto disk onto the remote machine, you type

***St_ore local file Example as remote file ExampleCR**

or ***St_ore local file Example as remote file DummyCR**

again depending on whether or not you want to use a different name. In this case, the message "[Old file]" or "[New file]" will not appear; remember, however, that storing a file on a file server ordinarily creates a new version rather than overwriting an existing file with the same name. The Store command does not accept file name patterns.

If you were not logged in at the time you started FTP, you will be asked for your user name and password when you do the first Retrieve or Store. FTP will save this information so that you won't have to provide it again until the next time you boot the Alto.

You can do as many Retrieve and Store commands as you want. When you are done, type

***Q_uit**

and you will be back talking to the Executive.

If you intend to do a lot of transfers to or from a directory other than your own, you can say

***Dir_ectory OtherDirCR**

to make <OtherDir> be the default directory for remote file names; this saves typing <OtherDir> in front of each name. Similarly, if you want to refer repeatedly to a sub-directory on an IFS file server, you can say, for example,

***Dir_ectory Jones>MemosCR**

Directory protections are ordinarily such that you cannot store into directories other than your own. However, if you know the password of some directory, you can *connect* to it by saying

***Con_nect to directory OtherDir Password xxxxCR**

which not only directs FTP's attention to that directory (just as does the Directory command) but also gives you complete access to it, as if you were its owner. The password is not displayed when you type it, of course. File servers generally have many so-called *files-only* directories that don't belong to any particular person but rather are for use by a group or project. The Connect command is the way you gain access to such a directory, particularly when you want to store files into it.

6.4 Listing and deleting files

You can get a list of the remote files that match a file name pattern with the List command; for example,

*List *bravoCR

or *List <Jones>Memos>*CR

It is quite slow, however, and there is no way to interrupt it except to SHIFT-SWAT out of FTP. You are better advised to obtain such information using the Chat program, to be described shortly.

To delete a remote file, type, for example,

*Del etc ExampleCR

After displaying the full name of the file, FTP will ask you to confirm or cancel your intentions. If there are multiple versions of the same file, the lowest-numbered version will be deleted.

6.5 Transferring files to and from another Alto

In addition to accessing a file server, FTP can communicate with any other Alto that is also running FTP. By this means, you can transfer files directly from one Alto disk to another.

To do this, boot the disk and start up FTP on the second Alto. Suppose its name is Banjo. If you don't know what its name is, you can use its network address, which FTP shows at the very top of the screen (e.g., "3#326#"). Now go to the first Alto, start up FTP, and tell it to connect to the second Alto simply by typing its name (much as you would type the name of a file server); for example,

*BanjoCR

or *3#326#CR

Now you may retrieve and store files as usual. Of course, in this case remote file names will not contain directories or versions (unless version numbering is enabled on the other Alto). Also, file name patterns will not be accepted in any context.

When you start FTP on an Alto, it is normally ready to act as a remote machine or *server* as just discussed. If you don't say anything special, it will allow any other machine to retrieve files, and to store new files, but not to overwrite existing files. You can change these defaults by starting FTP with

>Ftp/x

where x can be: -S (no server) to prevent any such transfers; Protected to allow retrieving only, but no writing; Overwrite to allow existing files to be overwritten. Any server activity is reported in the server window at the top of the screen.

6.6 Access via Chat

The File Transfer Protocol (the means by which FTP communicates with a file server) limits itself to the basic set of operations already described. There is an escape mechanism that lets you get at some important additional file server facilities directly, using a program called Chat. Chat uses the Alto display to simulate a traditional, "dumb" computer terminal, and thereby enables you to talk directly to Executive programs running in the file server machines themselves.

Maxc, being a general-purpose time-sharing system, has a large array of commands and programs; only the ones directly concerned with Maxc's role as a file server are described here (a few additional facilities are presented in section 6.8). The IFS Executive's command repertoire is limited to operations on files. Most of the commands described here are the same on IFS and Maxc; however, there are a few differences which you should note carefully if you use both systems.

To initiate a conversation with the Executive in a file server named, say, Ivy, just type

>Chat IvyCR

If all goes well, you will see the message "Connected to:" followed by some numbers at the top of the screen, a message from the server's Executive immediately below that, and "@" at the left margin prompting you for type-in. If Chat has trouble getting connected, it will tell you its problem after trying for a few seconds. This usually means that the server is broken; you might try again in a few minutes.

The next step involves logging into the server Executive. Chat may do this for you automatically, depending on the version of Chat you are using and the file server you are accessing. In this case, if you have forgotten to login to your Alto, Chat will first ask you for your account name and password. Otherwise, you must type

@Login (user) name (password) passwordCR

When the server types more than a screenful at you, it will pause after every screenful and "ring the bell", which causes Chat to display a large DING at the top of the screen. After you have had a chance to read the screen, striking any key on the keyboard will get the server to produce the next screenful. If you type ahead, this feature is suppressed.

Whatever the server Executive is doing, you can force it to stop by typing C^c. On Maxc, you may have to type C^c several times in quick succession to get it to stop.

When you are finished talking to the server Executive, type

@LogoutCR

(or "Quit" if the server is an IFS). This will terminate your Chat session and return control to your own Alto's Executive. If you are connected to Maxc, Chat won't terminate until about two minutes after you log out; to give control back to the Alto Executive immediately, type SHIFT-SWAT.

If the file server is an IFS, you will be logged out automatically if you don't type anything for two minutes. This is because IFS can service only a small number of users (currently five) at once; the automatic logout is intended to prevent IFS from being tied up by users who aren't doing anything useful.

Chat keeps a record of your conversation on a file called Chat.scratchScript. You can read it with Bravo after a Chat session, just to see what happened, or perhaps to copy things out of it into other files, print it, or whatever. There are two funny things about this file which you need to know about:

The file is not erased when you start a new conversation. Instead, the typescript of the new conversation starts at the beginning of the file and continues for as long as the conversation lasted. The end of the conversation is marked by the characters <=> after which you may see the remnants of previous conversations.

The typescript file is only 20,000 characters long. If your conversation is longer than that, the typescript will wrap around to the beginning. It is possible to make the file larger by editing the [CHAT] section of the user profile (the file User.cm) in the obvious way.

You can also initiate a Chat-like conversation with a file server while you are running FTP. At the bottom of FTP's screen is a "Chat window" (actually labelled "User Telnet"), in which you can talk directly with a file server's Executive in much the same manner as you do with Chat. You can move the blinking cursor down into the Chat window by striking the unmarked key to the right of right-SHIFT; to get back to the middle window, strike the unmarked key to the right of RETURN. In the Chat window, after typing the server name followed by CR, you can log in and do whatever you want. This window isn't very large and doesn't offer all the conveniences of Chat itself, but at times it is nice to be able to switch very quickly between transferring files and giving commands to the server Executive. You can make the Chat window larger when you start up FTP by typing FTP/-S, which prevents the FTP server from being started and thereby eliminates the upper (server) window.

6.7 Server Executive commands

You type commands to IFS and to Maxc in more-or-less the same way (except for those commands that have different names on the two systems); however, the responses from IFS and Maxc are usually somewhat different. The examples below illustrate IFS's responses. You may type "?" at any point to obtain a brief explanation of what you are expected to type in next. Maxc normally does not display the remainder of abbreviated commands or the explanatory text in parentheses; however, you can force it to do so by terminating fields you type in with ESC rather than space.

To generate a list of the names of all your files, type

on IFS: @ListCR

on Maxc: @DirCR

To list only those files matching some file name pattern, say, "Activity.*", type

*on IFS: @List(files) Activity.*CR*

*on Maxc: @Dir Activity.*CR*

To list another user's directory, type

*on IFS: @List(files) <OtherUser>*CR*

on Maxc: @Dir <OtherUser>.*CR*

Caution: note carefully that the command names are different on IFS and Maxc. Worse, there is also a List command on Maxc, but with an entirely different meaning (it causes hardcopy to be generated). Be careful!

You can obtain more detailed information about your files (length, date written, etc.) by typing a comma immediately before the CR; for example,

on IFS: @List(files) Activity.,CR*

on Maxc: @Dir Activity.,CR*

At this point, the server Executive will type "@@" at the left margin and permit you to type in one or more *sub-commands* that modify the action of the main command; in the case of List and Dir, subcommands are used to specify what information you wish to see about the files. Some of these are:

<u>@@TypeCR</u>	file type and byte size
<u>@@SizeCR</u>	size in pages
<u>@@LengthCR</u>	length in bytes
<u>@@CreationCR</u>	date of file creation
<u>@@WriteCR</u>	date on which the file was last written
<u>@@ReadCR</u>	date on which the file was last read
<u>@@TimesCR</u>	times as well as dates
<u>@@AuthorCR</u>	name of user who created the file
<u>@@VerboseCR</u>	same as Type Size Write Read Author
<u>@@EverythingCR</u>	everything known about the file

After you have typed in one or more sub-commands, type just CR in response to the "@@" prompt. The server will now perform the main command and list out the files with the information you requested. The columns of printout will be aligned correctly only if the font Chat is using is a fixed-width font, such as Gachal2 or Gachal0.

You can delete one or several files (or all files matching some file name pattern), just as on the Alto, with

@Delete file1 file2 ... CR

On IFS, the server Executive will now print out each file name and ask you to confirm your intention to delete it (type Yes or No); this is because once a file is deleted it is gone forever. On Maxc, all the files are deleted immediately without further confirmation; however, there is an Undelete command that you can use immediately afterward if you change your mind.

To delete all old versions of files (i.e., all but the highest-numbered version of each file), type

on IFS: @Delete *CR (*note the comma*)

@@Keep 1CR

@@CR

on Maxc: @DelverCR

 Delete oldest? YesCR

 Delete 2nd newest? YesCR

 File(s): CR

It is a good idea to do this fairly frequently, since old versions of files can pile up and waste a lot of space.

To find out how much space you are using on the file server, type

@DskStatCR

One IFS or Maxc page is equivalent to about four Alto pages. You will notice that you also have a *disk limit* which is the maximum number of pages you are permitted to use on the file server at one time. If you exceed your disk limit, the server won't let you store any more files until you first delete some existing ones to get you below your disk limit. To get your limit changed, consult your local support staff.

You can direct your attention to some other directory by typing

@Connect (to directory) OtherDir (password) passwordCR

just as in FIP. You may omit the password when connecting back to your own directory, or when connecting to a directory belonging to a project of which you are a member.

For the sake of security, it is a good idea to change your password occasionally (say, once a year). To do this, type

@Change Password (of directory) name (old password) xxx (new password) yyyCR

where *name* is your account name. The new password should be six or more characters long and unpronounceable (this is not enforced, however). Note that, contrary to normal practice, the new password *does* print out when you type it; this is so that if you make a typing mistake you will be able to see it.

6.8 About Maxc

Maxc's Executive is thoroughly documented in its own manual, which was written primarily for programmers and contains a large amount of information not needed by casual users of Maxc. The next few paragraphs document Maxc facilities likely to be of interest to Alto non-programmers. If you don't use Maxc you may skip this section.

If you have a file on Maxc in Press format (see section 7.1 for an explanation of file formats), you can tell Maxc to print it directly by issuing the "Press" command. This is documented in section 7.5.

Maxc provides facilities for *archiving* files onto magnetic tape, where the cost of storing them is negligible. You can get an archived file back within one day.

To archive one or several files, type

@Archive File file1 file2 ...CR

(Note that the command name consists of the two words "Archive File"; after that you should type the names of the files you want to archive.) The files will be archived onto tape within a day or

two. After this has been done, they will be deleted from the disk automatically, and you will get a message notifying you that the archiving has been done.

Maxc keeps track of your archived files in an *archive directory* which you can list exactly like your regular Maxc directory, using the Interrogate command rather than the Directory command; for example,

@Interrogate *.bravoCR

If the listing is of just one file, Maxc will ask you whether or not you want it retrieved from the tape. If you say Yes, the file will appear on your Maxc directory within a day, and you will get a message to that effect. It should be noted that the Interrogate command has some peculiarities not shared by other commands; in particular, you sometimes have to type an extra CR at the end of the command in order to get it to do anything.

Because Maxc's disk capacity is fairly small relative to the number of users who have Maxc accounts, the disk occasionally becomes full and it becomes necessary for a *forced archive* to be performed in order to make some space available. In a forced archive, all files that haven't been referenced (retrieved, printed, or whatever) in the past 90 days are written onto tape and deleted. You will be notified when any of your files are archived for this reason, and the procedure for getting them back is the same as given above.

7. Printing

The subject of printing is somewhat complicated because of the large number of variables involved. To begin with, there are many different *programs* that you can use to prepare documents for printing—Bravo, Draw, Markup, Sil, etc. Then there are various *file formats* defining the representation of documents stored in files—Bravo-format, Press-format, plain text, etc. Finally, there are several different types of *printers*—Dover, Sequoia, Slot/3100, Pimlico, etc.

This section first presents some introductory information on programs, file formats, and printers. Following that are a few of the most common procedures for printing documents.

7.1 Programs and file formats

Each of the programs you can use to prepare documents deals with information in a particular format. Bravo deals with text interspersed with special information about looks (fonts, paragraphs, etc.), and document files written by Bravo's Put command are in *Bravo format*, which only Bravo can read. The same can be said about Draw, Sil, and a number of other programs.

In order to be printed, a document generally has to be in *Press format*, which is a file format designed principally for representing printed pages. It follows that to print a document that is in some other format you must first convert it to Press format.

Programs that have their own special document formats also provide facilities for generating documents in Press format. For example, when you are using Bravo and you issue the Hardcopy command, Bravo first converts the document you are working on into Press format (this is why Hardcopy takes so long), then sends the resulting Press file to a printer. By using the File option of the Hardcopy command, you can tell Bravo to write the Press file on your disk rather than sending it to a printer.

The important point is this: a given document can be represented in several different formats. When you issue the Put command in Bravo you store the current document as a Bravo-format file, but when you issue the Hardcopy command with the File option you store the same document as a Press-format file. But while these are two different representations of the same document, only the Bravo-format file can be read back into Bravo (using the Get command), and only the Press-format file can be sent to a printer. *You can't read a Press file back into Bravo.* This is why it is important to choose file names in such a way as to identify their formats—the extension .Bravo to identify Bravo-format files, .Press for Press-format, .Draw for Draw-format, etc.

There are a few programs that deal exclusively with Press files. Markup is an illustrator that can both read and write Press files. PressEdit is a program used to manipulate Press files in various ways, such as combining several small Press files to create a single large Press-format document.

7.2 Printing servers

To print a document you send it through the network to a *printing server*, which is an Alto connected (usually) to some Xerographic printing device. The means by which you do this depend both on what kind of document you have and what type of printing server you are sending it to.

There is a bewildering array of names you will hear associated with printing servers. These names fall into three categories. There are *family names* that identify the type of printing device attached to the server Alto—a Dover is based on a Xerox 7000 copier, a Sequoia on a 3100, a Pimlico on a 6500, etc. Then there are names for different types of *printing software* used on the server Alto to control the printing device—Spruce and Press are the principal ones in use at present. Finally there are names identifying *individual printing servers*—the names by which you specify which particular

server is to print your documents. Examples of names of printing servers in Palo Alto are Clover, Menlo, Daisy, and Turkey.

To complicate matters further, while every printing server accepts Press files to print, there are different kinds of Press files and not all printers are capable of printing every kind of Press file. A Press file can contain a wide variety of information: text, straight lines, smooth curves, raster-scanned images, and even color. Obviously, if you send a Press file containing color to a black-and-white printing server you will not obtain the results you might desire; but there are other restrictions as well, most arising from technical considerations that will not be discussed here. As a rule, however, most printing servers will attempt to print any Press file as best they can and will tell you about whatever difficulties they may have encountered.

Here are some brief descriptions of the types of printing servers presently in use.

Dover is the predominant work-horse printer. It is based on a Xerox 7000 copier and is capable of continuous high-volume output (one page per second). It is best at printing documents consisting primarily of text, though it has limited capabilities for printing simple illustrations such as those produced by Sil. More complicated graphics (e.g., curves produced by Draw or raster-scanned images produced by Markup) can only be printed crudely if at all. Also, Dover prints pictures containing large solid black areas very poorly.

Sequoia is a smaller and slower printer, based on a Xerox 3100 copier. It can print more complex pictures than can Dover, and it prints solid black areas very well.

Pimlico is a Xerox 6500-based color printer. TC-200 is based on a Xerox Telecopier. Versatec is an electrostatic printer, some models of which can print on very large sheets of paper. Slot/3100 is similar to Sequoia. All these printers are capable of printing any kind of Press file (with the exception, of course, that color information is ignored by black-and-white printers).

Certain Press files contain copies of graphical information in two or more alternate forms. For example, curved lines produced by Draw are represented in the Press file both by mathematical descriptions and by Alto screen-resolution bit maps. Printing servers that understand the mathematical descriptions will use them to produce smooth curves, whereas servers that don't understand the mathematical descriptions—principally Dover and Sequoia—will print the bit maps instead, producing rather crude curves.

7.3 Fonts

There is a large array of *fonts* available for printing text. All fonts are named according to a standard convention:

family-name point-size face

The *family-name* describes the overall style of the font; e.g., TimesRoman or Helvetica. The *point-size* specifies the height of the font in points (one point is 1/72 inch). The *face*, if present, describes one or more additional properties of the font: B for bold, L for light, I for italic, C for condensed, and E for expanded. For example, Helvetica10B is a font in the Helvetica family, 10 points high, bold-face; TimesRoman12BI is TimesRoman, 12 points high, bold-face, italic.

A Press file containing text includes the *names* of the fonts to be used to print the text, but does not include any information about what the characters actually look like. Rather, each printing server maintains the printing representation of the set of fonts used most commonly by users of that printer. Many servers have a limit on the number of different fonts they can keep (related to the size of the disk attached to the server Alto). If you try to print a Press file containing text in a font the printing server doesn't know about, the server will substitute some other font and will tell you about this.

Programs such as Bravo that format text according to how wide the individual characters are obtain

the necessary information from a standard *widths* file named Fonts.widths, which must be present on your Alto disk. Additionally, in order to display text on the Alto screen you must have the appropriate *screen fonts* contained in files named *font-name.al*. You can usually obtain these files from the <Alto>/fonts directory in your file server. See section 4.6 of the Bravo manual for further information on how Bravo deals with fonts.

Here are a few fonts that most printing servers know about. There are samples of some of them at the end of the Bravo manual.

<i>Family</i>	<i>Point-sizes</i>	<i>Faces (aside from normal)</i>
TimesRoman	8, 10, 12	B, I, BI
Helvetica	7, 8, 10, 12 6, 18	B, I, BI B
Gacha	8, 10, 12	I
Cream	10, 12	B, I, BI
Math	8, 10	
Hippo	8, 10	
Arrows	10	

You should consult your support staff to find out what fonts are available on your local printing servers.

7.4 Printing from your Alto

After you first initialize your disk and before you attempt to print anything, you must edit your user profile (file User.cm) to declare the name of the printing server you intend to use regularly. See section 2.4 of the Bravo manual. All programs except Laurel that generate hardcopy look in User.cm to find out where to send Press files to be printed. For Laurel, you must also edit your Laurel profile, file Laurel.profile, described in section 3.6 of the Laurel manual.

Generating hardcopy directly from Bravo is easy: you just issue Bravo's Hardcopy command, described in section 2.4 of the Bravo manual. If instead of just printing hardcopy you wish to distribute a document on-line (say, by storing it on a file server so as to make it available to other Alto users), then rather than distributing the Bravo-format document you should make a Press-format file and distribute that. To do this, use the File option of the Hardcopy command and specify a file name with extension .Press.

Once you have a Press file on your disk (having either created it yourself or retrieved it from a file server), you can send it to your printing server using the Empress program. If you just type

>Empress filenameCR

one copy of the document will be printed by your default printing server. If you want more copies or you want to print on a different printing server, you can use the /C and /H switches; for example,

>Empress 3/C Menlo/H filenameCR

will cause three copies of the document to be printed by the printing server named Menlo.

To print a file that is in some other format, say Draw or Sil, you must first create a Press-format version of that file. The means by which you do this are described in the appropriate manual, i.e., in the Draw manual for Draw-format files, the Sil manual for Sil-format files, etc. Once you have a Press file you can print it using Empress as just explained.

There is an additional file format called *plain-text*. Basically it is a text file containing no font information and no formatting. You can create a plain-text file using Bravo if you start with an empty window and never type any CTRL-CR's or looks, but do type ordinary CR's at the ends of lines. A non-programmer is unlikely to encounter plain-text files very often; but, for example, User.cm,

Laurel.profile, and Executive command files (see section 8.2) are plain-text files, as are documentation files with extension .tty that you obtain from a file server (section 9.2).

You can print a plain-text file using Bravo Hardcopy, but there is an easier and much faster way: just type

>Empress filenameCR

Empress will discover that the file is plain-text rather than in Press format, and will convert it into Press format before transmitting it to the printing server. Empress will normally use a single font, Gacha8, for this purpose. You can change this to something else by editing the [HARDCOPY] section of User.cm to include a line such as

FONT: TimesRoman 10 B

The foregoing procedures for sending Press files to printing servers apply to those printers that run in *server mode*, i.e., that wait for someone to send them a Press file over the network and automatically print any files they receive. There are some printers that do not operate in server mode, usually because they cannot safely be left to run unattended. To print a document on one of these printers, you have to go to that printer's Alto, use FTP to retrieve the Press file you want to print, and type

>Press Print filenameCR

There are sometimes additional operating procedures which you will find posted near the printer.

7.5 Printing from Maxc

If you want to print a Press document or a plain-text file that is stored on Maxc, you can retrieve it to your Alto using FTP and send it to your printing server using Empress, as already explained. However, you can alternatively tell Maxc to send the file directly to the printer.

Before you do this, you must tell Maxc the name of your printing server. You do this by creating (with Bravo) a plain-text file containing the single statement

PDEVICE *server*CR

where *server* is the name of your printing server. Then Put onto file DocGen.prt, Quit, and use FTP to transfer DocGen.prt to your directory on Maxc.

Whenever you want to print a document that is stored on Maxc, you should connect to Maxc's server Executive using Chat and issue the command

@Press filenameCR

where *filename* is the name of a Press-format file (extension .press) or a plain-text file (e.g., extension .tty). You cannot use the Press command to print other kinds of files—in particular, Bravo-format files.

If your DocGen.prt file on Maxc also contains the line
REPORT YCR

then Maxc will attempt to notify you as soon as it has actually sent the file to the printer. Maxc will display a message on your screen if you are still connected to Maxc's server Executive at the time; otherwise, it will send you a message that you will receive next time you run Laurel.

7.6 PressEdit

You can compose the various parts of a document with Bravo, Markup, Draw, Sil, or other programs that produce Press files, and then put together the complete document with a program called PressEdit. You can also use PressEdit to extract pages from an existing Press file. PressEdit has some other features, some of which are documented here and the remaining ones in the Alto Subsystems manual.

The use of PressEdit for assembling documents has one major advantage: the resulting complete document can be left on a file server for printing by anyone who needs a copy. If you are producing a document for large-scale printing outside, on the other hand, it is probably easier to assemble it by hand than to go through all this ritual. One restriction you should be aware of is that every printing server has a limit on the size of Press file that it can handle (this is principally a function of the capacity of the disk connected to the server Alto). Most printing servers can handle Press files up to about 50 pages long (printed pages, not Alto disk pages), though some can handle documents substantially larger than that. If you have a very large document, you should distribute it as several Press files, each containing no more than 50 pages.

The simplest use of PressEdit is to append together two or more smaller Press files to create a single large one. For example,

>PressEdit Manual.press ← Chapter1.press Chapter2.press Chapter3.pressCR

creates Manual.press by concatenating the three other Press files in the order given. Be sure to type a space both before and after the "`←`". You should also remember that the new Press file will occupy as much disk space as the three existing Press files combined; check that you have enough free disk pages before you start.

You can copy selected pages out of a Press file and put them into a new Press file by a command such as

>PressEdit Short.press ← Long.press 3 6 10 to 14 18 to 22CR

This extracts pages 3, 6, 10 through 14, and 18 through 22 from Long.press and puts them in Short.press. Note that the numbers refer to consecutive pages in the source Press file, counting from 1, and have nothing to do with any page numbers that actually appear on the pages themselves.

The concatenate and extract operations may be combined in one command to produce a document with pages from two or more source documents interleaved. This is particularly useful for inserting illustration pages into text documents. For example,

>PressEdit Report.press ← Text.press 1 to 3 Figures.press 1 Text.press 4 to 8 Figures.press 2 Text.press 9 to 14CR

This produces a document consisting of pages 1 through 14 of Text.press, with page 1 of Figures.press inserted between pages 3 and 4 of Text.press and page 2 of Figures.press between pages 8 and 9.

If you want to make a document that has pages containing both text and illustrations, there are two ways to *merge* selected pages of different Press files. Both techniques are unfortunately rather cumbersome. The first method involves interleaving the pages of the text and illustration Press files, as just described, and then using Markup to copy material from one page to another in the resulting file. This procedure is documented in section 4 of the Markup manual. It is very slow and requires a lot of manual labor, and it does not always work for illustrations produced by any program besides Markup.

The second technique requires you first to put special marks in the text and illustration Press files to show how you want the illustrations to be positioned. You then run PressEdit, which merges the source Press files automatically to produce the desired document.

Each illustration must be contained in a separate, one-page Press file. Somewhere in the illustration must appear an "arrow" consisting of the following piece of text:

`<= =<<`

There should be no spaces or other characters either before or within this piece of text; you must position the arrow using the text positioning facilities of the illustrator you are using.

The main text document must also contain an arrow to show the position of every illustration. Inside each arrow must appear the name of the Press file that contains the illustration to be inserted there; for example,

<= =<Fig3.press<

Again, there must be no spaces or other characters either before or within the arrow; you must position it by setting the left margin appropriately and possibly by using the vertical tab feature—see sections 3.2 and 3.6 of the Bravo manual.

Having prepared all the Press files, you merge them by typing a command of the form
>PressEdit/M Document.press ← Text.press Fig1.press Fig2.press Fig3.pressCR

After the "←" you must type the name of the main text file followed by the names of all the illustration files you are inserting into the final document. You may list the illustration files in any order, and if you are inserting a particular illustration into more than one place in the final document, you need type its name only once.

Now, each time PressEdit encounters an arrow in the text document, it merges into that page the illustration contained in the Press file whose name is inside the arrow. PressEdit positions the illustration on the page by aligning the two arrows, then removes the arrows.

PressEdit also has a facility for adding fonts to Press files. This is useful primarily when you want to edit the Press file with Markup and add text in some font that does not already appear in the file. For example, to add fonts Logo24 and Helvetica12 to file Example.press, type

>PressEdit Example.press ← Example.press Logo24/F Helvetica12/FCR

The next time you read Example.press into Markup, the new fonts will appear in Markup's font menu in addition to the ones that were there before.

8. Other things

This section describes various facilities and procedures that you will probably find useful at some point. Browse through them now just so you know where to find them.

8.1 Copy and Rename

To copy one file to another, say

>Copy new ← oldCR *don't leave out the spaces*

The "←" is to remind you of the direction the copying is done.

To change the name of a file, say

>Rename new ← oldCR *don't leave out the spaces*

or: >Rename old newCR

There must not already be a file called *new*. However, if *old* and *new* differ only in capitalization, Rename may be used to change the capitalization.

8.2 Command files

If you have a sequence of Executive commands that you wish to execute repeatedly, you may put them into a *command file*, then invoke the command file at any later time. To create a command file, use Bravo to enter the exact commands that you would issue to the Executive, then Put the document onto a file with the extension ".cm", the standard extension for command files.

To execute a command file named, say, Cleanup.cm, type

>@Cleanup@CR

The Executive will display the first command in the command file, perform the command, and then go on to the next command, continuing until it has executed all the commands in the file. To abort execution of a command file, type C^c, which will stop it at the end of the current command, or SHIFT-SWAT, which will stop it immediately.

Actually, you may substitute the contents of a command file for any part of an Executive command line. For example, if file ListOffFiles.cm consists entirely of the text "Alpha Beta Gamma Delta" (with no CR at the end), and you type

>Delete @ListOffFiles@ EpsilonCR

the effect will be exactly the same as if you had typed

>Delete Alpha Beta Gamma Delta EpsilonCR

A number of commonly-used programs, most notably FTP, are capable of accepting their own commands from the Executive command line used to invoke them. Normally, when you start up FTP simply by typing

>FtpCR

FTP then expects you to type commands to its own keyboard command interpreter. But if you start it up with, say,

>Ftp Ivy Store/C Memol.bravo Report2.pressCR

FTP will make a connection to the Ivy file server, store the files Memol.bravo and Report2.press on your Ivy directory, and return control to the Executive, with no further action on your part, except to type in your password if you haven't already logged in.

This capability may be used in conjunction with command files to permit you to deal with large groups of files all at once. For example, if you have a set of Bravo documents comprising one large report, you can create a command file, say, Report.cm, containing the names of those files, then issue the command

>Ftp Ivy Store/C @Report.cm@CR

to transfer a complete, consistent set of those files to a file server.

You should be aware that the language used to control FTP from the command line is *not* the same as that used to control it from the keyboard, though it is similar. You should read section 4 of the FTP manual before attempting to create FTP command lines.

8.3 Dump files

The Executive's Dump command gives you a way to package up a number of files into a single, so-called *dump file*. You can then transport the dump file around as a unit, and later recover one, a few, or all of the files from it using the Load command. This is especially useful in maintaining consistent sets of related files.

To make a dump file, type

>Dump Alpha.dm file1 file2 ... CR

Here "Alpha.dm" is the name of the dump file; by convention it has the extension "dm." You can list as many files as you want to be dumped. Often the * feature of the Executive is useful here, and of course you may obtain the list of files from a command file.

To get files back from a dump file, type

>Load/v Alpha.dmcR

You will get a list of the files in Alpha.dm, and after each one you will be asked whether you want it loaded or not. If you leave out the /v all the files which don't already exist will be loaded; if you say /c instead, all the files will be loaded whether or not they are already on your disk.

The FTP program has facilities for accessing dump files on a file server: you may transport a collection of files on your disk to or from a remote dump file without ever having to put the dump file on your disk. That is, the command

>Ftp Ivy Dump/C Alpha.dm file1 file2 ... CR

will package together files *file1*, *file2*, etc., and store them as Alpha.dm *in the file server*, not on your Alto disk. Similarly,

>Ftp Ivy Load/C Alpha.dmcR

will access Alpha.dm on Ivy and load the constituent files onto your Alto disk. You will probably find that this is more convenient than using the Executive's Dump and Load commands. See sections 3 and 4 of the FTP manual for complete information.

8.4 Neptune and DDS

The Neptune program provides convenient facilities for managing files on your Alto disk. Basically, it displays the names of all the files on your disk in a window that you can scroll just as in Bravo, and it permits you to specify operations on individual files simply by pointing at their names with the cursor. It can delete files, display the contents of text files, and (if your Alto has two disk drives) copy files from one disk to another.

The Neptune manual is included at the end of the Alto User's Handbook. If you read sections 1 through 4 of that manual you will know enough to start using Neptune.

There is another program called DDS which is considerably more powerful than Neptune and provides a number of useful capabilities that Neptune lacks. However, it is rather slow and takes up a great deal of space on your disk, so it is not included on the BASIC NON-PROGRAMMER'S DISK. You should try it out and see whether its features are valuable to you. The DDS manual appears as part of the Alto Subsystems manual (see section 9.2).

8.5 Illustrators

There are currently three major programs for drawing pictures on the Alto:

Markup: good for pictures involving images, free-hand drawing or painting. Markup is also useful for adding pictures to a text document produced by Bravo; these pictures can come from Draw or Sil, or they can be drawn by Markup itself.

Draw: good for pictures which contain lines, curves and text.

Sil: good for forms and pictures with only horizontal and vertical lines. For such pictures it is much faster than either Markup or Draw.

Manuals for Markup and IDraw are included in later sections of the Alto User's Handbook. Sil is also suitable for general use; unfortunately, the present Sil documentation is rather terse and is oriented principally toward users of the Design Automation system, of which Sil is a part. You can obtain this documentation, such as it is, by printing <Sil>SilManual.press.

8.6 CopyDisk

The simplest use of the CopyDisk program is copying the contents of one disk pack to another on an Alto equipped with two disk drives; it was described in section 2. CopyDisk can also copy the contents of a disk pack from one Alto to another over the Ethernet. To use it in this mode, you need two Altos; in the example below they are called Banjo and Flash.

Put the disk you want to copy *from* into one Alto (say, Banjo), and use the NetExec to invoke CopyDisk (see section 3.4). Put the disk you want to write *onto* into the other Alto (Flash), and start CopyDisk on that Alto also. You will type all commands on Flash, i.e., the Alto containing the disk you want to write *onto*.

You should now go through the following dialogue:

Copy to: DP0CR

Copying onto DP0 will destroy its old contents.

Are you sure this is what you want to do? [Confirm] Yes

Are you still sure? [Confirm] Yes

In the above example, in response to "Copy from:" you type the name of the *other* Alto (the one you are copying *from*) in square brackets, followed immediately by "DP0" (with no intervening spaces). If you don't know the name of that Alto, you can instead type its Ethernet address (displayed in the black bar on that Alto's screen) followed by a #. In response to "Copy to:" you type simply "DP0", meaning the disk pack in drive 0 of the Alto on whose keyboard you are typing.

After you have confirmed your intentions, the copy should proceed. When CopyDisk is done, if all went well you will see the message "Done. [Banjo]DP0 and DP0 are identical" followed by the "*" prompt. You may now type QuitCR to each Alto.

8.7 Version numbers

There is an optional *version number* facility that permits you to keep multiple versions of a particular file on your Alto disk without having to invent a different name for each one. This can be particularly valuable when you are making a number of successive changes to a document but want to keep earlier versions around in case you change your mind. Files stored on file servers always have version numbers, but use of version numbers on Alto disks is optional.

Unfortunately, the version number capability has not been integrated fully into all Alto programs (in particular, the Laurel message system). Furthermore, the relatively small amount of storage available on an Alto disk makes wholesale maintenance of multiple versions impractical. For these reasons, the version number facility is *disabled* on the BASIC NON-PROGRAMMER'S DISK. If you wish to use it, you may enable it by doing an Install, asking for the "long installation dialogue", and answering the questions appropriately.

If the version number facility is enabled, a file name may end with an exclamation point followed by a number; for example, "Alto.Manual!4" is version 4 of the file Alto.Manual. The basic rule for version numbers is this:

When you read a file, you get the one with the largest version number (the *current* version), unless you include the version number you want in the file name.

When you write onto a file for which the current version is n , a new version $n+1$ is created, and becomes the current version, unless you include the version number in the file name. Furthermore, if version $n-1$ was around, it gets deleted, so that just two versions of the file are kept, the current one (with the largest version number) and the next earlier version. The number of versions kept may be changed at Install time.

For example, if version 4 is the current version of the file Alto.Manual, there will probably be "Alto.Manual!4" and "Alto.Manual!3" around. If you write onto "Alto.Manual" (c.g. by doing a Put from Bravo), "AltoManual!3" will disappear, and "Alto.Manual!5" will appear with the new information on it. "Alto.Manual!4" will still be around unchanged, so you can get the old version back from there if you need it. On the other hand, if you write onto "Alto.Manual!4", that file will be changed, and no new versions will be created.

If a file name doesn't have a version number, most programs will not make any new versions, but will just write on the single version. Bravo is an exception; it always makes new versions if the version number facility is enabled.

9. Software distribution and documentation

Alto software and documentation are publicly available from file servers. Most file servers maintain duplicate copies of common files. In Palo Alto, most software and documentation of interest to non-programmers is stored on Maxc; in other places, on the local IFS. You should consult your support staff for the exact maintenance policies used in your organization.

9.1 Obtaining new software releases

When new versions of the various programs are released, they are normally announced by messages to all registered Alto users. You can obtain a new version of a service called, say, Alpha as follows:

If the release announcement includes instructions for installing the new version of the program, follow those instructions. Otherwise:

Using FTP, attempt to retrieve <Alto>Alpha.cm from your file server. If this succeeds, leave FTP and type to the Executive

>@Alpha.cm@CR

This will cause FTP to be invoked again, some files to be transferred from your file server, and perhaps some other activity. When everything settles down, you will have the new version.

If there is no <Alto>Alpha.cm, retrieve <Alto>Alpha.run. This will be the new version of the program. You don't have to do anything else.

It is a good idea to keep fairly up-to-date. New versions of programs are sometimes released to fix serious bugs or to reflect important changes in operating procedures. If you run into trouble while running an obsolete version of some program, you are unlikely to receive much help or sympathy from the program's maintainer or from your local support group.

The best way to obtain a complete set of new software, and clean up your disk at the same time, is to obtain a fresh disk, initialize it from the BASIC NON-PROGRAMMER'S DISK as described in section 2, and then use FTP to transfer all the files you want to keep from your old disk to the new one, as described in section 6.5. If you have an Alto with two disk drives, you can put the old disk in one drive and the new one in the other, then use Neptune to copy files between disks. See sections 1 through 4 of the Neptune manual.

An alternative way to make a BASIC NON-PROGRAMMER'S DISK is to put the disk you want to initialize into an Alto, boot the NetExec as described in section 3.4, then type

>NewOSCR

You will get a fresh version of the operating system, which will ask you if you want to Install. Say Yes, ask for the "long installation dialogue", and say that you want to erase a disk. After a minute or so, you will have a clean disk with nothing on it except the Executive and FTP. Use FTP to retrieve the file <Alto>NewNpDisk.cm from your file server. Then type

>@NewNpDiskCR

This will automatically transfer all the needed files from the file server, and do any other necessary initialization. It takes about 20 minutes, and puts a significant load on the file server, so use this procedure only when you can't find the BASIC NON-PROGRAMMER'S DISK. During the operation, there will be an automatic Install of the operating system; answer its questions appropriately. You will have to type your name and password at various points in the procedure. There will also be an automatic initialization of Bravo, and you should do a Quit when it is finished.

9.2 Documentation

Documentation for all the standard Alto software can be found in the <AltoDocs> directory on Maxc and other file servers. As a rule, each major piece of documentation appears as a Press file which you can obtain and print by means of the procedures explained previously (section 7). Short documents are available as files with the extension "tty"; these are plain text files that you can transfer to your Alto and read with Bravo or print with Empress.

To see what is available, you can Chat to your file server and type

on IFS: @List <AltoDocs>*.press <AltoDocs>*.tty

on Maxc: @Dir <AltoDocs>*.press <AltoDocs>*.tty

Here is a short guide to on-line documentation likely to be of interest to non-programmers but not already contained in the Alto User's Handbook. All are stored on the <AltoDocs> directory except where noted otherwise.

Alto Subsystems, files Subsystems1.press and Subsystems2.press.

This manual, which is about 150 pages long, contains documentation for a large number of Alto programs. Included is comprehensive information on Chat, CopyDisk, Empress, and the Executive, which are only partially described in the Alto User's Handbook. An additional program of interest to non-programmers is DDS, which enables you to display and manage your Alto disk's file directory in ways much superior to Neptune.

Alto Subsystems Catalog, files SubsystemsCatalog.press.

This is a summary of Alto subsystems, organized by function.

Sil, Analyze, Gobble, Build Reference Manual, file <Sil>SilManual.press.

Sil is an illustrator specialized for very rapid composition and editing of pictures consisting of straight lines and text. It is part of the Design Automation system for digital logic development, but Sil is useful in its own right as a general-purpose illustrator.

Printing at Palo Alto, file Printing.press.

This is a comprehensive summary of Press printing facilities, formats, and programs. Despite its title, it is applicable more-or-less everywhere.

How to Use IFS, file <IFS>HowToUse.press.

This is a complete user's manual for IFS, presenting a fair amount of material not covered in the Alto User's Handbook.

The Alto User's Primer, file AltoUsersPrimer.press.

This document contains some introductory material on Alto hardware and software. It is oriented towards newcomers to the Whole Alto World, and its main purpose is to describe what exists and how to get it.

Whole Alto World Newsletter.

This is a monthly newsletter that serves as a vehicle for communication of Alto-related information among Alto users throughout Xerox. Each month's edition is stored as WAWNews m -yy.press; for example, WAWNews8-78.press is the August 1978 edition.

5

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: bravomanual.press

Creation date: 14-Dec-79 11:44:36 PST

For: falis

32 total sheets = 31 pages, 1 copy.

Problems encountered:

Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files

Bravo Manual

Table of Contents

Preface	32
1. Introduction	33
2. Basic features	34
2.1 Moving around in a document	34
2.2 Changing the text	35
2.3 Filing a document	37
2.4 Hardcopy	38
2.5 Miscellaneous	39
3. Formatting	40
3.1 Making pretty characters	40
Looks during typing	
3.2 Paragraphs	41
Hints	
3.3 Formatting style	44
Emphasis	
Section headings	
Leading	
Indenting	
Offsets up and down	
3.4 Forms	46
3.5 White space and tabs	47
3.6 Page formatting	48
Page numbers	
Margins	
Multiple-column printing	
Line numbers	
Headings	
4. Other things	52
4.1 Some useful features	52
4.2 Windows	53
4.3 If Bravo breaks	54
4.4 Arithmetic	55
4.5 Other useful features	56
Buffers	
Partial Substitution	
Control and special characters	
4.6 The user profile and fonts	58
4.7 Startup and quit macros	59
4.8 Press and Diablo hardcopy	60
Samples of standard fonts	61
Summary	62

Preface

This manual describes the Bravo system for creating, reading and changing text documents on the Alto. It is supposed to be readable by people who do not have previous experience with computers. You should read the first four sections of the Non-Programmers Guide to the Alto before starting to read this manual.

You will find that things are a lot clearer (I hope) if you try to *learn by doing*. Try out the things described here as you read.

Material in small type, like this, deals with fine points and may be skipped on first, or even second, reading.

This manual is written on the assumption that you have the user profile, fonts and other Bravo-related material from the BASIC NON-PROGRAMMER'S DISK. If this is not the case, some of the things which depend on that stuff will not work the same way.

There is a one-page summary of Bravo at the end of this manual. It is intended as a memory-jogger, not as a complete specification of how all the commands work.

Bravo was designed by Butler Lampson and Charles Simonyi, and implemented mainly by Tom Malloy, with substantial contributions from Carol Hankins, Greg Kusnick, Kate Rosenbloom and Bob Shur.

1. Introduction

Bravo is the standard Alto system for creating, editing and printing documents containing text. It can handle formatted text, but it doesn't know how to handle pictures or drawings; for these you should use Draw, Markup or Sil.

When you start up Bravo (do it now, by saying Bravo/eCR to the Executive), you will see two windows on the screen, separated by a heavy horizontal bar. The top one contains three lines with some useful introductory information; it is called the *system window*. The bottom one contains a copy of the material you are reading, which was put there because of the "/e" you typed to the Executive. If you had omitted the "/e", as you do when using Bravo normally, the bottom window would be empty, except for a single triangular endmark which indicates the end of a document. In the bar separating the two windows is the name of the document in the lower window.

As you do things in Bravo, the first two lines of the system window will give you various useful pieces of information which may help you to understand what is going on and to decide what you should do next. Usually, the top line tells you what you can do next, and the second line tells you what you just did, and whether anything went wrong in doing it. *Make a habit of looking at these two lines while you are learning Bravo, and wherever you are unsure of what is happening.*

No matter what is going on in Bravo, you can stop it and get back to a neutral state by hitting the DEL key. You can leave Bravo and get back to the Executive by typing

QuitCR

The characters which you type (Q and CR) are underlined in this example; the characters which are not underlined are typed by Bravo. This convention is used throughout the manual. Notice that you only type the first character of the Quit command; this is true for all the Bravo commands.

Each Bravo window (except the top one) contains a *document* which you can read and change. Usually you read the document from a file when you start Bravo, and write it back onto a file after you have finished changing it. Later, you will find out how to do this (section 2.3). It is possible to have several windows, each containing a document; this too is explained later on (section 4.2).

Bravo is controlled partly from the keyboard and partly from the *mouse*, the small white object with three black buttons which sits to the right of the keyboard. As you push the mouse around on your working surface, a *cursor* moves around on the screen. Pushing the mouse to the left moves the cursor to the left, pushing the mouse up (away from you) moves the cursor up; and so forth. You should practice moving the mouse around so that the cursor moves to various parts of the screen.

The three buttons on the mouse are called RED (the top or left-most one, depending on what kind of mouse you have), YELLOW (the middle one) and BLUE (the bottom or right-most one). They have different functions depending on where the cursor is on the screen and what shape it has. Don't push any buttons yet.

Mouse lore:

You will find that the mouse works better if you hold it so that it bears some of the weight of your hand.

If the cursor doesn't move smoothly when the mouse is moving, try turning the mouse upside down and spinning the ball in the middle with your finger until the cursor does move smoothly as the ball moves. If this doesn't help, your mouse is broken; get it fixed.

You can pick the mouse up and move it over on your work surface if you find that it isn't positioned conveniently. For instance, if you find the mouse running into the keyboard when you try to move the cursor to the left edge of the screen, just pick the mouse up and set it down further to the right.

2. Basic features

This section describes the minimum set of things you have to know in order to do any useful work with Bravo. When you have finished this section, you can read the other parts of the manual as you need the information.

2.1 Moving around in a document

Move the cursor to the left edge of the screen and a little bit below the heavy black bar. Notice that it appears as a double-headed arrow. It will keep this shape as long as you stay near the left edge, in a region called the *scroll bar*. If you move it too far right, the shape will change. Keep the cursor in the scroll bar for the moment.

Now push down the RED (top or left) button and hold it down. Notice that the cursor changes to a heavy upward arrow. This indicates that when you let the button go, the line opposite the cursor will be moved to the top of the window. Try it. This is called *scrolling* the document up.

Next push down the BLUE (bottom or right) button and hold it down. Now the arrow points down, indicating that when you let the button go, the top line on the screen will be moved down to where the cursor is. Try it. This operation takes a few seconds, so don't get impatient. Practice scrolling the document up and down until you feel comfortable with it. It is useful to know that if you don't move the mouse, scrolling with RED and BLUE are symmetrical operations: one reverses the effect of the other.

You may have noticed that the text on the screen doesn't fill up the window, but that more text appears when you scroll up. The reason for this is that in addition to space on the screen, Bravo needs space inside itself (in the Alto's memory) to display lines of text on the screen. When a line has only a few characters, it doesn't take up much internal space, but when it runs all the way across the page, like the lines in this document, it takes a lot of internal space. When Bravo runs out of internal space, it stops displaying text and leaves the rest of the window blank. You can tell that there is more text in the document (i.e., that you aren't seeing the end), because when Bravo gets to the end it displays a triangular *endmark* as the very last thing to mark the end. If you don't see the *endmark* at the bottom of the displayed text, you can be sure that there is more text in the document which isn't being displayed.

If you keep the cursor in the scroll bar, near the left edge, and hold down YELLOW (the middle mouse button), you will see the cursor change into a striped right-pointing arrowhead. Think of it as a thumb, and the entire left edge of the window as the pages of a closed book, corresponding to your whole document (*not* just to what is displayed). If you stick the thumbnail into the book and flip it open, you will find yourself someplace in the book. If the thumb is near the middle, you will be about in the middle. If it is all the way at the top, you will be at the beginning; if all the way at the bottom, you will be at the end.

The tip of the arrowhead acts like the thumbnail, and letting go of YELLOW is like flipping open the book. You will also see another striped arrow, enclosed in a box. This one is called the *bookmark*; it points to your current location in the document. After you let up YELLOW, if you hold it down again without moving the mouse, the thumbnail and the bookmark should coincide exactly, making a solid arrowhead; this happens because the thumbing operation moved the document exactly to the place indicated by the thumbnail. To move forward a little, push the thumbnail down a little below the bookmark and thumb again; to move back, push the thumbnail up a little above the bookmark. To get to the beginning, push the thumbnail up until the arrowhead overlaps slightly the horizontal bar at the top of the window. Try thumbing your way through the document until you feel comfortable with it. Also try thumbing and then scrolling up and down.

2.2 Changing the text

In order to make a change in the text of your document, you have to:
say where you want the change made, by making a *selection*;
say what you want done, by giving a *command*.

You always make the selection first, then give the command. If you change your mind about where you want the change made, you can always make another selection. Making a selection is just like pointing with a pencil: it doesn't have any effect on the document. Only commands can change the document. You never have to worry about getting rid of a selection, since it never does any harm. If you make a selection, and then give a command that doesn't require any selection, that is perfectly all right; the needless selection will be ignored.

You make selections by pointing with the mouse and pushing one of the buttons. To try this out, move the cursor into the region of the screen where the text of the document is displayed. Notice that the cursor is displayed as an arrow which points up and slightly to the left. Point the arrow at a character (any character) in the document, and click RED. The character you pointed at should be underlined; if it is, you have just selected it. If it isn't, look nearby and see if some other character is underlined. If you find one, then that is the one Bravo thought you were pointing at. Experiment until you feel confident that you can point easily at characters.

You should note that each selection erases the previous one; there is only one selection at a time, and it is the most recent one. Also, you can make a selection at any time, except when you are in the middle of a command. Once you have started a command, you must either finish it normally, or abort it by striking DEL, before you can make another selection.

Something useful to know: if you hold RED down, you can move the cursor around and the selection will follow it. The selection won't freeze until you release RED (or move the cursor out of the text area). Try this too.

Now try a selection using YELLOW instead of RED. Notice that instead of underlining a character, Bravo now underlines a whole word. A word is defined as consecutive letters and digits, or consecutive punctuation characters. For convenience, apostrophe is counted as a letter. Also, a number containing a decimal point is a single word.

There is one more thing to learn about selecting text: how to select more than one character or one word. To do this, first select a character with RED. Then point to another character and click BLUE; Bravo will underline all the characters between the one you selected with RED and the one you pointed at with BLUE. This is called *extending* the selection. Try holding down BLUE and moving the cursor around. The selection will change continuously so that it includes the characters between the one you originally selected with RED and the one you are pointing at now. As before, when you let up the button, the selection will freeze. You can change the extension as many times as you want by using BLUE over and over; Bravo will remember the original selection you made with RED until you make another one.

Finally, try selecting a word with YELLOW and then using BLUE to extend the selection. Notice that the end of the selection will be a word also. To select the entire document, issue the Everything command.

Space, TAB and carriage return (CR) characters in the document simply appear as white space on the screen, just as they do on paper. You can, however, select them like any other characters. Try it. You will notice that not all the white space on the screen can be selected; in fact, the space on a line after a CR, and the space to the left of the left margin, cannot be selected. Bravo's handling of white space is discussed in detail in section 3.5.

Now that you know how to say where you want a change made, it's time to make a change. Select something (for instance, a word). Now type D (for Delete). The word you selected is *deleted* from the document, and the selection moves over to the character after the original selection. The rest of the text is adjusted to make up for the deleted material; if necessary some words may be brought up from the next line to fill up the one which contained the deleted material.

You can *undo* the deletion by typing U (for undo). Try it; you will see the stuff you deleted reappear, and it will be selected again, just as it was before you deleted it. Do several deletions, followed by undos, until you are sure you know what will happen. Try deleting larger pieces of text by extending your selections. Be sure not to move the selection between doing the Delete and the Undo.

Delete and Undo are *commands*. Like all Bravo commands, they are given by typing just the first letter of the command name. You can type the letter in either upper or lower case.

To add new text, select something in front of which you want the new text to go (if you want it to go at the very end of the document, you can select the *endmark*). Then type I (for Insert). You will see that a blinking caret appears just before the selection. This marks the place where the new text will go. Anything you type will appear where the caret is, and as you type each character, the caret will move over to make room for it. Try typing a few characters, and notice that the rest of the text is automatically rearranged to make room for the new stuff.

If you strike the wrong key while typing, you can erase the mistake by striking the BS key (on the right side of the keyboard). You can erase as many typed characters as you like using BS. You can also use A^c (hold down the CTRL key and type A) to erase a character; it works just like BS, and may be more convenient to type with your left hand, if your right hand is on the mouse. To erase typing on a larger scale, you can use W^c (hold down the CTRL key and type W) to erase a word and its following spaces or punctuation characters. When you have typed as much as you care to, hit ESC to finish the insert. Notice that the caret disappears, and that the inserted material is selected. You can undo the insertion with Undo. Then you can undo the undo and get the insertion back. Try it.

Sometimes it is more convenient to insert *after* a selection, rather than *before*. You can do this with the Append command (remember that you just type the A). Except for where the new material goes, Append is exactly like Insert.

If you want to change one word into another, or correct a typo, you have to delete some text and insert other text in its place. This can be done by a Delete followed by an Insert, but it is more convenient to use the Replace command, which combines these two functions into one. Replace can also be undone.

Whenever Bravo first displays the blinking caret, you can insert a *copy* of some existing text rather than typing in new text. You do this by making another selection, called a *copy selection*, instead of typing. The copy selection is made exactly like an ordinary selection, and you can even use the scroll bar to move around in the document in order to find the text you want to copy. You can distinguish a copy selection from an ordinary one by its dotted underline, which contrasts with the solid underline of an ordinary selection.

You can change your copy selection as many times as you like. When you are satisfied with it, type ESC, and a copy of the copy selection will be inserted in place of the blinking caret. You can't do anything else while you are making a copy selection, except to scroll the document.

A copy selection can be used to move text from one place to another: first copy the text, and then delete the original.

There is one more useful thing to know about insertion. If you just type an ESC for an insertion, without making a copy selection or typing anything else, a copy of the last thing you inserted or deleted will be inserted. This is called repeating or *defaulting* an insertion; it is very convenient for inserting the same thing in several places, e.g., a dollar sign in front of several numbers. It also gives you another way to move text: first delete it, and then insert it in its new place by selecting the new place and typing Insert followed by ESC.

You now know all three ways of doing an insertion: *typing* the text, *selecting* some existing text to be copied, or *defaulting* the previous insertion by simply typing ESC. These three ways of inserting text can be used whenever a Bravo command needs some text. You will see many references to "inserting text" as you read on.

Before going on to learn anything more about Bravo, you should practice the Delete, Insert, Append and Replace commands, and copy selections, until they are quite familiar.

2.3 Filing a document

Whether you use Bravo simply to read or browse through a document, or to create or change it, you will need to fetch the document from a file before starting, and to file it away again afterwards if you have changed it. This section tells you how to do these things.

To fetch a document from a file, give the Get command. You will see the blinking insertion caret appear in the heavy black bar above the window. Insert the text of the file name, usually by typing it in, and ending it with an ESC just as for any other insertion. The document will appear in the document window, and there will be a note in the system window telling you how long it is. A Get will erase the old contents of the window, if any.

To file a document away, give the Put command, and type the file name as you did for Get. If the name you want is already in the black bar, you can just type ESC to default the name. It is also possible to edit the file name in the black bar, exactly like an ordinary document. Put always files away the entire document, regardless of what the selection is; when it is done, you will see a note which tells you how long it is. Bravo turns most of the screen black while executing a Put; this makes the Put run faster. Do not be alarmed at this.

Warning: If you make some changes to your document and then attempt to Quit from Bravo without having done a Put, Bravo will warn you that the document has not been filed and will ask you whether you still wish to Quit. If you want to save the document, strike DEL to cancel the Quit command, then file the document using Put. If you want to quit without saving the document, type Yes. If you do this, you will lose any changes you have made to the document. If this does happen to you, read section 4.3 on replaying to see if you can still be saved.

If you Get a document from a file and Put it back on the same file, Bravo will save the original on a *backup* file. Normally the backup file's name will be the name of the original file followed by a "\$". The backup file is sometimes useful if you discover that some of the changes you made are not to your liking after all. If you have enabled version numbers at Install time (not recommended), the backup file will be the old version of the file from which you did the Get, and Bravo will make a new version for the Put (see section 8.7 of the Alto Non-programmer's Guide for a discussion of file versions).

You can do an "unformatted Get" with the Z^c command (type Zc instead of Get): this treats the formatting information at the end of each paragraph as ordinary text. The main use of Z^c is for patching up a file which has been damaged by hardware failure or cosmic rays. In particular, if Bravo refuses to Get the file because "End of file not in Bravo format", you can usually correct the problem by doing an unformatted Get of the file, deleting the last line or two, and Putting it back. Then Quit, restart Bravo and try again to Get the file.

2.4 Hardcopy

Printed copies of a document may be obtained using the Hardcopy command. Before using Hardcopy for the *first* time, you must tell Bravo the name of the printer you intend to use regularly. To do this, Get the file User.cm. In that document, you will find an entry that looks like this:

```
[HARDCOPY]
PREFERREDFORMAT: PRESS
PRESS: Name-of-your-Press-printer
```

Replace the words "Name-of-your-Press-printer" with the name of the printer you intend to use (every printer has a registered name such as Clover, Menlo, or Daisy). Then Put the document, Quit, and type Bravo/iCR to the Alto Executive.

To print one copy of the document you are editing, simply give the Hardcopy command followed by CR. This will print the entire document, regardless of what the selection is. While doing the hardcopy, Bravo displays in the cursor a count (modulo 10) of the number of pages it has processed; hardcopy takes about 8 seconds per full page, like this one. After sending the document to the printer, Bravo will report success. If there is a problem, Bravo will leave a note in the system window. If the printer is not responding, Bravo will leave a note to that effect, and keep trying. You can abort the Hardcopy by typing DEL, as always.

The hardcopy may fail for several reasons. If there is an EITP error, trying again will usually work. If the problem is that there is a character in your document which is in a font for which there is no printable representation, Bravo leaves one of the offending characters selected, and puts it at the top of the screen. You can try again after modifying the looks of the selected character. If you have a page with so many different fonts that it exceeds the capacity of the printer, Bravo leaves the first character of the page selected and at the top of the window. There is no remedy for this problem except to simplify the offending page. See section 4.6 for more information about fonts.

You may want more than one copy of a document. The Hardcopy command has an option called Copies, which allows you to specify the number of copies you want; you type in the number, and it will appear in the leftmost buffer in the system window, much like a file name. You must give the Copies option right after the Hardcopy command, every time you want more than one copy.

If you compare the hardcopy of your document with Bravo's display, you will see that although the text is identical, the hardcopy has more words on each line, so that the two versions look quite different. In order to see a nearly exact facsimile of the hardcopy on the screen, you can give the command

Look hardcopy (note the lower-case h)

You are now in *hardcopy mode* on the screen. Until further instructed, Bravo will represent the printed version of your document as faithfully as it can, by positioning each character on the screen within one-half screen dot (about .007 inches) of its position in the final hardcopy. The screen representation is 10% larger than the printed one. To turn off the hardcopy simulation, type

Look Hardcopy (note the upper-case H)

You can edit normally in hardcopy mode. In fact, if your document contains tables whose appearance is critical to you, it is advisable to stay in this mode, because in the normal mode text will take up much more space on the screen than it will in the final hardcopy (if you have such tables, you should also read section 3.5 on white space and tabs). In hardcopy mode it is also possible to see exactly where lines will be broken, so that you can insert hyphens by hand if necessary.

Bravo provides a number of facilities for controlling page formatting, which you can read about in section 3.6.

The Hardcopy command has options for printing on the Diablo printer, and for producing a Press file which can be combined with drawings into a larger printable document, or sent to a file server for public distribution. These are described in section 4.8.

2.5 Miscellaneous

As you edit, Bravo keeps track of the changes you make to the document. In doing this, Bravo consumes space in the Alto memory. During a long editing session, it is possible to consume all the available space, in which case Bravo will leave a warning note ("Core storage getting low") in the system window, and will refuse to execute any more editing commands. If this happens, you should Put your document onto a file immediately, and then Quit, restart Bravo, and Get the document back from the file. Now you can continue with another editing session.

When you have finished editing one document and have filed it away, you can Get another file, and continue working. If you are making extensive changes, however, it is better to Quit and restart Bravo when you start to work on a new document. If you do this, you are less likely to provoke a bug in Bravo, and you will be able to recover from a crash with the replay feature (section 4.3) much more quickly.

The maximum size of a Bravo document is 65,536 characters. Whenever Bravo Gets or Puts a document, it leaves a note telling you how long the document was. When your document has reached 65,536 characters, you won't be able to add any more text, and peculiar things may occur if you do try to add more text. It is a good idea to split the document into two parts well before this happens. To encourage you to do this, Bravo will flash the screen and display a warning message after every command if the length of the document exceeds 60,000 characters.

If you type a character which has no printable representation, Bravo will display it as a black rectangle. The best thing to do with such a character is to delete it.

Depending on exactly what Bravo is doing, the amount of text it can display on the screen will vary. You can always get the maximum amount of text displayed by doing a scrolling operation; if you scroll up with the cursor at the top of the scroll bar, the text won't move, and Bravo will just display as much more as it can. If you then give a command, some of the text may disappear from the screen, but you can always get it to reappear by doing another scrolling operation.

Bravo keeps copies within itself of information in your user profile (file User.cm; see section 4.6) and in various files on your disk: font files (named *.al and Fonts.Widths), and the files containing the Bravo system and its temporary storage (named Bravo.*). It refreshes these copies whenever you start it up with

>Bravo/icr

This is called *initializing* Bravo. It is necessary to initialize whenever you get a new version of Bravo or the Alto Operating System and when you change your user profile or any font file. Initializing is just like starting Bravo up normally, except that it takes about a minute. *If you are in any doubt* about whether something has changed since the last time you initialized Bravo, or if your Bravo is crashing with messages which refer to disk or file errors, *you should initialize Bravo by starting it with Bravo/i.*

You now know enough to edit unformatted documents. Take a rest.

3. Formatting

This section describes the Bravo facilities for creating formatted text and pages. If you are not interested in formatting, you don't have to read it. If you are interested, be sure to read sections 3.3 and 3.4, where you will find a lot of good advice.

Bravo normally describes character sizes and distances on the page in *points*. A point is a unit of distance used in the printing industry; there are 72 points per inch. Thus 36 points is 1/2 inch, and 18 points is 1/4 inch. In many cases, you can also specify distances in inches or centimeters, as described in section 3.2.

3.1 Making pretty characters

Bravo allows you to say how you want your text printed: in italics or bold face, underlined, in various sizes and type styles, superscripted or subscripted, etc. You can change the way existing text is printed, or you can say how you want the characters to appear as you are typing them in. We will begin by describing how to change the looks of existing text.

First, select the text you want to mess with. Then give the Look command. This command has a large number of options, each specified by a single character, which is sometimes followed by some additional information:

<u>bold</u>	<u>SHIFT B</u> to un-bold
<u>italic</u>	<u>SHIFT I</u> to un-italicize
<u>—</u> to underline	<u>SHIFT —</u> to remove the underline
<u>←</u> to subscript (text down 4 pts)	<u>Down 0 ESC</u> to remove subscript or
<u>SHIFT ←</u> or <u>↑</u> to superscript (up 4 pts)	superscript
<u>0</u> to <u>9</u> to set the typeface	
<u>visible</u> to display spaces, tabs, and CRS	<u>SHIFT V</u> to stop this.
<u>Down</u> followed by a distance (see below) to move the text down that distance, relative to the baseline. Subscript is Down 4.	<u>Up</u> followed by a distance to move the text up. Superscript is Up 4.
<u>CLR</u> (the blank key to the right of BS on an Alto-I, or the key labelled BW on an Alto-II) to restore the standard looks: font 0; not bold, italic, underlined, visible, graphic, up, or down.	

The typeface is usually called the *font*. For Bravo, each different size of the same style is a different font, but bold and italic are considered to be in the same font. The choice of fonts is specified by your *user profile* in a way which is described later (in section 4.6), but the standard choice provided on the basic non-programmer's disk is:

- 0 Times Roman, 10 pt. This is the standard font.
- 1 Times Roman, 8 pt.
- 2 XEROX logo (only the capital letters E O R and X)
- 3 Math, 10 pt. A large set of mathematical symbols. No bold or italics on hardcopy.
- 4 Greek, 10 pt. No bold or italics on hardcopy.
- 5 Times Roman, 12 pt.
- 6 Helvetica, 10 pt.
- 7 Helvetica, 8 pt.
- 8 Gacha, 10 pt. This is a fixed-pitch font.

- 9 Helvetica, 18 pt. The bold-face version of this font is especially good for making view-graphs.

You will find tables at the end of this manual which give the correspondence between ordinary characters and the Math and Greek fonts, and some samples of the various fonts.

There is another Look option which is very convenient. It is Look Same, followed by a copy selection. In this case, what is copied is the looks, rather than the characters. This is the way to get one piece of text to print in the same style as another piece.

Like most commands, Look can be repeated with ESC. This is useful if you want to change the looks of several pieces of text in the same way. You can also undo a Look with Undo.

You can find out what the looks of a character are by selecting it and giving the Look ? command. Bravo will tell you (in the system window) all the looks of the selected character. You may have to scroll the system window up in order to see all the looks.

LOOKS DURING TYPING

When you start typing, the looks which will be attached to the characters you type are set to the looks of

- the first character of the selection if the command is Insert or Replace;
- the last character of the selection if the command is Append;
- the standard looks otherwise.

To change the looks while you are typing text, use the CTRL key instead of the Look command: hold down CTRL and type the look you want. The only things described above which you can't do this way are Look Up and Down; you can get the standard superscript and subscript offsets with ↑ and ←, though. To restore the standard looks, you can just strike the CLR key; it is not necessary to use CTRL in this case.

3.2 Paragraphs

In addition to changing the looks of the characters, you can also change the shape of the text: the margins, space between lines, justification, centering, etc. The Bravo facilities for doing this are based on the idea of a *paragraph*.

A paragraph in Bravo is all the text between two CTRL-CR characters. You can tell when you have one by selecting it. To do this, move the cursor into the *line bar*, which is between the scroll bar on the far left, and the text area. You can tell that you are in the line bar, because the cursor will appear as a rightward-pointing arrow. Once you are in the line bar, use the YELLOW button to select a paragraph. Note that the cursor changes to a paragraph symbol; it keeps this shape as long as the selection is a paragraph.

The CTRL-CR that ends a paragraph carries the paragraph looks described below. It can also carry character looks, and if you are setting up a standard paragraph, it is a good idea to attach to its CTRL-CR the character looks which you want as the standard ones for the paragraph. Thus, for example, the CTRL-CR for a standard heading like the one at the start of this section would carry the italic look. Of course, this is just a convenience, and not essential; you can always set the character looks during typein as described above, e.g. by typing ic for italics.

If the text at the end of a paragraph is in a font smaller than the standard one, as this one is, the CTRL-CR ending the paragraph should carry the same font looks. Otherwise, the inter-line spacing of the paragraph may appear uneven.

The **YELLOW** button selects exactly *one* paragraph, so by looking at what is underlined you can tell where the paragraph starts and ends. Note that the second CTRL-CR (the one that ends the paragraph) is counted as part of the paragraph; the first CTRL-CR is part of the previous paragraph. You can use **BLUE** to extend the selection to several paragraphs.

To merge two paragraphs into one, just delete the CTRL-CR that separates them. You will probably want to replace it with a couple of spaces, or maybe with an ordinary CR. To break one paragraph into two, insert a CTRL-CR; it is just like any other character, except that you can't backspace over it.

If you select a paragraph and then give an Append, Insert or Replace command, a blank paragraph with the same looks as the selected one will be created for you to type into.

To change the looks of a paragraph, you can use some more sub-cases of the Look command. Select the paragraph (or any text in it) first, and then say Look, followed by:

<u>c</u> enter; turns off justification	<u>S</u> HIFT <u>C</u> to stop centering
<u>j</u> ustify (even right margin); turns off centering	<u>S</u> HIFT <u>J</u> to stop justifying
<u>n</u> ested to indent the whole paragraph (36 pts, or 1/2 inch, more)	<u>S</u> HIFT <u>N</u> to un-indent
<u>o</u> pen up more white space in front of the paragraph (12 pts, or 1/6 inch, more)	<u>S</u> HIFT <u>O</u> to close up the white space
<u>q</u> to open up half as much more white space in front of the paragraph as Open does (6 pts more)	<u>S</u> HIFT <u>Q</u> to close up the white space

All of these can be invoked during type-in; hold down the CTRL key and strike the appropriate key, just as you do for character looks.

In the following Look cases, *d* is a *distance* on the page, which can be specified in several different ways, as described below. Distances are measured from the left edge of the paper (except for Up and Down, which measure from the baseline of the line of text). These looks cannot be used during type-in.

Left *d* to set the left margin. The default is 85 points, or about 1.2 inches from the left edge of the paper.

First *d* to set the left margin of the first line. Use this to control indenting or un-indenting of the first line. A Look Left cancels a Look First, since it sets the left margin for all the lines of the paragraph.

Paragraph *d* to set the left margin of all the lines except the first. A Look Left cancels a Look Paragraph, since it sets the left margin for all the lines of the paragraph.

Right *d* to set the right margin. The default is 527 points. Since an 8.5" x 11" page is 612 points wide, this results in 85 points, or 1.18", of white space on the right. Thus, the default margins center the text on the page.

X *n* to set the space or *leading* between lines. The leading should be at least 1 point (as it is in this document) to avoid a squashed effect. If you want a less dense appearance, try larger leadings. The default is 6pt, which gives double spaced text.

Y *n* to set the leading in front of the paragraph. The default is 12pt, which gives a blank line between paragraphs. Look open increases the paragraph leading by 12 pts, and Look q increases it by half that, or 6 pts. On hardcopy, both line and paragraph leading are suppressed for the first line of a page or column. Leading must be less than 64 points.

Here are the ways to specify the distance. Try them out until you are quite comfortable with them.

By typing a number in one of the following forms:

123 or 123pt	a distance in <i>points</i> . A point is a printer's unit equal to 1/72 of an inch. A number without a decimal point and with no explicit units is assumed to be in points.
1.71, 1.71in or 1.71"	a distance in inches. A number with a decimal point and no explicit units is assumed to be in inches.
4.34cm	a distance in centimeters.
a sequence of blanks	a distance equal to the width of that many blanks.

By typing a number *n*, as above, preceded by + or -. The distance specified by *n* is added to, or subtracted from, the current value of the look being changed. Thus, to indent a paragraph by an additional one-half inch, type Look Left +.5 ESC.

By using BLUE to point to a place on the screen. The horizontal position of the place you point at is displayed in the system window. If you hold down BLUE and move around, the displayed position is updated continuously.

By using RED to select a character. The horizontal position of the left edge of the character is displayed in the system window.

By typing \ (back-slash, not /), which displays a default value for the look being changed in the system window.

By just typing ESC, which uses the value already in the leftmost buffer of the system window.

You can select, point or default as many times as you want, just as with an ordinary copy selection. Then you can type a number, if you like. When the leftmost buffer in the system window has the value you want, type ESC to complete the command. Of course, if you get disgusted you can always type DEL to cancel the whole thing. Note that pointing is a convenient way to measure horizontal positions on the page.

Look All, followed by a copy selection, will copy all the paragraph looks of the paragraph in which the copy selection is made, to the paragraph containing the current selection. Note that Look All copies paragraph looks whereas Look Same copies character looks.

If a paragraph is selected (using YELLOW in the line bar; the cursor will be a paragraph symbol when a paragraph is selected), the Look ? command will display the paragraph looks in the system window; if the selection is not a paragraph, the command displays character looks, as described in the previous section. You may have to scroll the system window to see all the information. Note that it appears in a buffer (see section 4.5) which is made current, and you can insert it into a document with a default insertion.

If you have a paragraph whose left margin is less than the default (normally 85 pts), any characters in the paragraph to the left of the default margin will fall off the left edge of the window and will not be displayed. Try setting a left margin to some values less than 85, and see how this works. You can change the setting of the left edge of the window, so as to make these characters visible on the screen, with the command

Window Edge *d* *d* is a distance, which must be typed in and cannot be obtained by pointing.

The distance *d* is the distance from the left edge of the page at which the left edge of the window should be set. It should be smaller than any paragraph left margin if you want to see all the characters on the left. For instance, if *d* is 0, the window edge will be at the paper edge; if the text has the usual 85 pt margin, this will result in 1.2" of white space in the window (in addition, of course, to the white space in the line and scroll bars). The default value for *d* is the default left margin.

HINTS

You can select several paragraphs (using BLUE to extend your selection) and apply the same Look command to all of them. You can change the looks of every paragraph in the document by doing an Everything to select the whole document before the Look. A Look command involving a distance of the form $+n$ or $-n$ adds or subtracts n from the look value for each selected paragraph. Thus, Look Left +5 ESC will indent each selected paragraph by five more points.

If you use several different formats (e.g., for section headings or for indented material) you can copy the formatting from an existing example of a particular style to a newly created one with Look All. Often it is convenient to put a set of sample paragraphs at the head of your document, each containing one line which explains what it is a sample of. Then you can split the window (as described in section 4.2) and have the samples readily available to copy from with Look All. This is highly preferable to entering all the new looks manually every time you switch to a new format.

An alternative technique for creating a new paragraph in a specific style is to select the paragraph before or after the point at which the new paragraph is to appear, then Append or Insert and make a copy selection of the desired sample paragraph. Now select the text of the newly-created paragraph, *not including the CTRL-CR at the end*, and Replace it with new text that you type in. This method copies both character looks and paragraph looks from the sample paragraph.

When you are setting up the format for a document, you should put a few blank paragraphs (just CTRL-CRS) at the end, and set the formatting on all of them to your standard format (it is convenient to do this by copying the formatting from a paragraph which already has your standard format). This might include indenting the first line of a paragraph, setting the leading, leaving space between paragraphs, justification, and even the font. Now when you add material to the document by inserting into one of these blank paragraphs, you will automatically pick up all of the formatting you have preset. As you type along, each time you use a CTRL-CR to start a new paragraph, it will acquire the same formatting as the old one.

3.3 *Formatting style*

This section is intended to provide you with some guidance in using all the different ways Bravo gives you for controlling the appearance of your document. Many of the rules are based on the customs of the printing industry. There are two advantages to following these customs:

- they are the result of many years of experimentation, during which many people have tried to find out what looks good on the page;
- readers are accustomed to seeing text presented in this style.

You will notice that some of the rules are contrary to the usual practice for preparing documents on a typewriter. There are good reasons for this: when you are printing with variable-pitch fonts, italics, boldface, justification, and precisely controlled leading, some of the things which work well for fixed-pitch, single-font documents are no longer appropriate.

EMPHASIS

Use *italics* for *emphasis* in text. You can also use *boldface*, but this is usually less desirable, and it is better to reserve *boldface* for words which play some special role, e.g., *begin* and *end* in computer programs. You should also use *italics* for the names of variables, e.g., "Suppose there are n items."

Don't use underlining for emphasis; it is not compatible with the use of *italics* and *boldface*. Use underlining only when you want a different *kind* of emphasis, e.g., to distinguish the characters a user types from the ones the machine types, as is done in this document.

Don't capitalize a whole word for emphasis. In fact, try not to capitalize a whole word at all; it usually looks terrible in a variable-pitch font because the capital letters are so much wider than the small ones. If you have words which you think should be set in capitals for some reason, try **SMALL CAPITALS**. In this example, the S and C were 10 point (font 0), the rest of the letters 8 point (font 1). Compare this with the appearance of **FULLY CAPITALIZED** words and you will see the point.

SECTION HEADINGS

In general, use left-justified rather than centered headings, and don't use all capitals, for the reasons just discussed. Here is a satisfactory list of styles for the headings of successively larger portions of your document:

smallest	<i>Italic</i>	18 pt paragraph leading (<u>Look Y 18</u> , or <u>Look q</u> if your standard leading is 12 pts).
next	Bold	24 pt paragraph leading (<u>Look Y 24</u> , or <u>Look o</u> if your standard leading is 12 pts).
largest	12 pt bold	36 pt paragraph leading (<u>Look Y 36</u> , or <u>Look o</u> twice if your standard leading is 12 pts).

Note that you can switch from the standard leading to the 1.5, 2 or 3 times standard leadings for headings during typein, using qc and qc. For the largest units, you can center the heading and/or use all caps instead of, or as well as, switching to a 12 pt font. It is best not to have more than three levels of heading, but you can extend to four or five levels using these tricks. Helvetica 18 bold (font 9 bold) is sometimes nice for chapter or document titles.

Use Look Keep 80 (see section 3.6) on headings to make sure that the heading doesn't end up all by itself at the bottom of a page.

LEADING

The standard printing fonts are designed in such a way that they need some extra space between the lines to avoid a cramped appearance. You put this space in with Look X, and you should use 1 pt for ordinary single-spaced text. If you want a less dense appearance, experiment with more leading. For double-spacing of the text, try Look X 6 (the default).

Use double spacing (Look o) between paragraphs. When you have indented material which is fairly short, try 6 pt leading (Look q), as in the example two paragraphs back. *Don't* use extra carriage returns to get blank space between paragraphs. However, the maximum leading you can specify is 63 points; if you need more (e.g., to leave space for a figure) you will have to put in blank paragraphs.

Note that both line and paragraph leading are suppressed for the first line of a page or column. The height of a line of text (in points) is equal to the point size of the largest font used in the line, provided there are no characters which have been superscripted, subscripted or offset with Look Up or Look Down. If any character in the line is offset Up, the minimum line height, *including* leading, is given by the font size of the character, plus its offset; i.e., characters offset Up are allowed to eat into the leading. If a character is offset Down, the largest such offset must be added to obtain the line height; i.e., characters offset Down are *not* allowed to eat into the leading.

INDENTING

Use Look nested to indent material, and Look Nested to cancel the indentation. Note that this also works when you are typing in. For example, if you type

CR^cn^c Here are three points: CR^cn^cFirst ..CR^cSecond ..CR^cThird ..CR^cN^cNow we continue ... the document will look like this:

Here are three points:

First ..
Second ..
Third ..

Now we continue ..

Use Look First if you want to indent the first line of a paragraph, rather than tabs. When you have a list of items, it is often nice to *un*indent the first line by about 15 pts, especially if the items are numbered. For example:

1. This paragraph was formatted with Look Left 120, Look First -15, in order to make the number hang out to the left.
2. To get the first word of the first line to line up with the left margin on subsequent lines, set a tab stop at that point (see section 3.5).
3. The easiest way to specify the position of the tab stop is to select the first character of the second line, using RED. In this case, of course, the stop is at 120.
4. Indented paragraphs sometimes look better balanced if the right margin is indented as well. Unfortunately, Look nested doesn't do this for you; you have to change the right margin yourself using Look Right d.

OFFSETS UP AND DOWN

Use the smallest offset you can get away with for subscripts and superscripts, since large offsets result in wide ugly spaces between the lines. The offset used by Look ↑ (superscript) and Look ← (subscript) are defined in your user profile (see section 4.6); the standard profile sets it to 4 pts.

3.4 Forms

Although Bravo will let you begin with a completely empty window and start typing into it, this is a bad practice and should be avoided. Instead, you should start out by Getting a *template* or *form* which will guide you in constructing the document you want.

An obvious example is a memo form, and you will find one on the file Form.Memo. Start Bravo, and Get Form.Memo into the window. You will see that it has spaces for the sender, receiver, date and subject, and that these are filled in with words which indicate what should go there. To fill in the form, select each of these words, and Replace it with the proper text. Then do the same with the MEMOBODY. When you are done, you have a completed memo which you can file under a suitable name using Put. Be careful not to Put the document back onto the file from which the form came. The best way to avoid doing this is to edit the file name in the black bar above the document immediately after Getting the form.

Your disk comes equipped with a few forms; you can see their names by typing form. TAB to the Executive. You should construct your own forms for other kinds of documents which you find yourself creating frequently. As you have seen in the description of Bravo's formatting features above, a form can contain a great deal of information in addition to standard text and spaces to be filled in. You will find that your life is easier and your work is more uniform and of higher quality if you use forms consistently, and take the trouble to carefully design a new one when necessary.

3.5 White space and tabs

When you type on a typewriter, you can get white space to appear between characters by typing spaces or TABs. You can get blank lines by typing carriage returns. In Bravo, you can do exactly the same things, with exactly the same results. Space, TAB and CR are characters which are in your document exactly like "a", "b" or "c". You can get Bravo to display them as special, visible characters by selecting the text in which you want to see them, and typing

Look visible (this must be a lower-case v).

To turn off the display and just see the usual white space, type

Look Visible (this must be an upper-case V).

Normally you don't have to type any CRs; Bravo will automatically end a line when there is no room for the next word. You can force a line to end by putting in a CR; this is appropriate when you want to control the layout of the text precisely, as in a table. Otherwise, don't put in CRs. You should use CTRL-CR to end a paragraph, as described in section 3.2.

Bravo allows you to set up to 14 tab stops, which are named by the digits 1-9 and the letters abcde. The tab stops are paragraph looks, just like the margins; hence they can be different for each paragraph. You can set a tab stop with the command

Look TAB t d

where *t* is a digit or one of the letters abcde, and *d* is a distance (see section 3.2).

When you strike the TAB key during typein, the caret moves to the next tab stop, just as it does on a typewriter, and a TAB character is added to the document. This TAB character is called a *plain-tab*, because it moves the caret to the next tab stop, not to a specific named tab stop.

For example, suppose you have a line like this:

Column 1 Column 2 Column 3

Tab stops 1, 2, and 3 are at 180, 265 and 400 points, and there is a plain-tab between each digit and the following C. If you now append some x's to the digit 1 to get past tab stop 1, the result will look like this:

Column 1xxxxxx

Column 2

Column 3

That is, the point to which a plain-tab jumps depends on the width of the preceding text. This can vary both when you change the text and when you switch to hardcopy mode; thus, the appearance of hardcopy may not match the screen if you are using plain-tabs.

You can turn a plain-tab into a *named-tab* by selecting it and issuing the command Look , t (Look comma *t*), where *t* is the name of a tab stop. A named TAB character will always make the following character print at the correspondingly named tab stop. If printing has already passed that tab stop, Bravo will start a new line, and display a heavy black rectangle at the end of the previous line, to warn you that something is wrong.

To continue the above example, suppose you name the first TAB 1 and the second 2. Now the result will look like this:

Column 1xxxxxx■

Column 2

Column 3

When you switch from normal display mode to hardcopy mode, there will usually be more white space occupied by the TAB (perhaps enough to permit printing all the text on one line), but everything will continue to be positioned horizontally in exactly the same way.

You can find out the name of a tab stop by selecting it and giving the **Look ?** command.

Caution: the Look comma command should be applied only to TAB characters. If applied to a character other than TAB, it will invoke some unsupported features for color printing.

For compatibility with old versions of Bravo, and with the programmer-oriented tab conventions of the Alto and Maxc, you can set unnamed or *plain* tab stops spaced at equal intervals with the command

Look TAB = d

where the distance *d* specifies the interval. If you don't set any tab stops, you get plain tab stops spaced at 36 pts (this parameter comes from the user profile, and can be changed; see section 4.6).

One final word about white space: Bravo has formatting features, described in the section on paragraphs above, which allow you to indent the first line of a paragraph, and to put blank space above a paragraph, without using spaces, TABS or extra CRs. It is good practice to use these features, since you can control the spacing much more precisely and don't have to worry about having extra characters cluttering up your document.

3.6 Page formatting

There are a number of features to help you in controlling the layout of your document on printed pages. Unlike the horizontal layout, the location of page breaks and the headings, page numbers etc. for the most part cannot be displayed on the screen. There is, however, a *page boundary* command which allows you to see on the screen where the page boundaries will appear in the hardcopy. The command is invoked by the LF key. It assumes that the first character of the current selection is the first character on a hardcopy page, and it moves the selection to the first line of the next page. By applying the page boundary command repeatedly, you can move through the document, page by page (or column by column, if your document profile specifies multiple columns; see below). Alternatively, if you know where one page break is (perhaps because of a control-L in the previous line; see below), you can start there. If you want to start at the beginning of the document, you can use the Everything command to make the first character of the document be the first character of the selection.

As a convenience, the page boundary command leaves the original selection at the top of one subwindow, and the first line of the next page as the third line of the next subwindow (which it creates if necessary). Among other things, this makes it easy to do some editing near the end of the page, and then reselect the beginning of the page and repeat the command. Try it.

Normally, Bravo will start a new page when it runs out of room on the current page, i.e., when the next line to be printed would intrude on the bottom margin, or at the beginning of a paragraph if the amount of space left before the bottom margin is less than the paragraph's keep value. You can force a page break by including a L^c (control-L) in the text; the line containing the L^c will stay on the same page, but the next line will start a new page. This character is displayed as a lower-case L with an over-bar. You can't type it in simply by holding down CTRL and typing L, but instead you can type an L followed by S^c. You do this *during an insertion*, not as a command. The L^c is treated just like any other character during editing.

You can also force a *paragraph* to start a new page by giving it a keep property of 11". If you want to position the paragraph precisely on the new page, give it a vertical tab property as well.

You can exercise some control over where page breaks occur with the command

Look Keep d *d* is a distance

This sets the paragraph property called *keep*, which has the following meaning. During hardcopy,

when printing of the paragraph is begun, the amount of space left on the page before the bottom margin must be at least the keep distance, or a new page will be started. For instance, by setting the keep of a heading to the total height of the heading (including its leading) plus the height of the first two lines of the next paragraph, plus the paragraph leading, you can ensure that the heading will never end up alone at the bottom of a page. Good values to use, with standard fonts and leading as in this document, are 40 pts on ordinary paragraphs and 80 pts on headings.

You can set the vertical position of a paragraph precisely on a page using the vertical tab property, which is set by the command

Look Z d *d* is a distance.

When a paragraph with a vertical tab is printed, its upper edge (including leading, if any) will be positioned at the vertical tab value, measured from the *bottom* of the page (i.e., use 10.5" to put it .5" from the top). Unlike a horizontal tab, which may start a new line, a vertical tab never starts a new page; instead, it may cause overprinting. Vertical tabs are useful for positioning headings and footnotes, and for precisely aligning text to meet some physical constraint, such as a pre-printed form or a window envelope. The first line of a paragraph with a vertical tab will be printed on the current page, even if it runs into the bottom margin (but not if the paragraph also has a keep property which forces it off the page).

Vertical tab and keep properties are not visible on the screen, but you can always use Look ? to find out whether a paragraph has them, and what their values are.

Note that both line and paragraph leading are suppressed for the first line of a page of column. If you want white space in front of such a line *l*, you can use vertical tabs, or introduce a blank line in front of line *l*, and adjust the leading of *l* to compensate for the height of the blank line.

The remaining aspects of page formatting can be controlled by an optional *document profile* which you can put at the very beginning of the document. The document profile is a sequence of paragraphs, each of which must have the *profile* property. This property is set and cleared by a Look command:

Look ; sets the profile property Look SHIFT ; clears it

A document profile has the following form (this one is the profile for this part of this manual):

```
Page Numbers: Yes X: 527 Y: -.5" First Page: 40 Not-on-first-page
Private Data Stamp: No X: 3.5" Y: -.6"
Columns: 1 Edge Margin: .6" Between Columns: .4"
Margins: Top: 1.3" Bottom: 1" Binding: -5
Line Numbers: No Modulus: 5 Page-relative First Line: 1
Odd Heading: Not-on-first-page
```

BRAVO MANUAL

Even Heading:

Section 3: Formatting

Any of the lines may be omitted, and in general any of the fields on a line may be omitted. Fields on a line are separated by one or more spaces. Distances, shown in inches in the example, may be given in points or centimeters, as described in section 3.2. X coordinates are from the left edge of the paper, Y coordinates from the bottom; negative coordinates are measured from the right edge or top of an 8.5" x 11" page. Bravo's measurements on the page are exact to less than .01". Actual printers, however, can make errors in positioning the text on the page of as much as .25" in any direction. These errors do not affect the *relative* positions of characters (e.g., the length of a line cannot be affected) but they can cause the text to shift around on the page as a whole.

We now proceed to explain the various options.

PAGE NUMBERS

The coordinates of the page number are the coordinates of the upper right corner of the number. You can add Roman to the line if you want Roman numerals for your page numbers, and Uppercase if you want the Roman numerals in upper case. If Not-on-first-page is present, the page number is not printed on the first page of the document. If First Page is not specified, it is assumed to be 1, and Not-on-first-page is also assumed for both page numbers and heading; i.e., there will be no page number or heading on page 1 in the default case.

Each page number's looks are copied from the first character of the heading printed on the same page, if there is one, or from the first character of the document otherwise.

The coordinates of the private data stamp are for its upper right corner. Do not use a private data stamp without proper authorization. You will need to supply a password on each hardcopy to get the private data stamp applied; see your laboratory manager to learn the password if you have a legitimate need.

MARGINS

The top margin specifies the amount of white space at the top of the page. The bottom margin specifies the minimum amount of white space at the bottom of the page; a line will start a new page if any part of it intrudes into the bottom margin. Exception: if a paragraph has a vertical tab, its first line will be printed without regard to the bottom margin, and it may be positioned without any regard to the top margin.

If Binding appears, it is assumed that the pages are eventually to be printed on both sides of the paper, with odd-numbered pages on the right side of the resulting double spreads. Page numbers of even pages will be reflected left-to-right; in the example, even page numbers will have their upper *left* corner at X: .5" Y: .5". The binding distance is the amount of extra margin to be supplied on the inner side of the page, which abuts the binding. This amount is added to all the X coordinates on odd pages and subtracted from all the X coordinates on even pages. For example, if you want 98 pt (1.36") outside margins and 72 pt (1") inside margins, use a left margin (Look Left) of 85 pt (the default), and a right margin (Look Right) of $612\text{ (8.5")} - 85 = 527$ (also the default) to center the text on the page. Then use a Binding of $72 - 85 = -13$. In general, the rule is:

$$\text{Let } d = (\text{desired outside margin} + \text{desired inside margin})/2$$

$$\text{Look Left} \quad d$$

$$\text{Look Right} \quad 612\text{ (8.5")} - d$$

$$\text{Binding:} \quad \text{desired inside margin} - d.$$

This rule will lead to a negative binding if the inside margin is less than the outside margin; that is perfectly all right.

MULTIPLE-COLUMN PRINTING

The columns line is relevant only for multiple-column pages. It says that the hardcopy should have the specified number of columns, with the nominal edge margin (at both edges) specified (.6" in the example), and the amount of space between columns specified (.4" in the example). If the number of columns in the example is changed to 2, the nominal horizontal layout of an odd page will be:

.6" edge margin; 3.45" text, column 1; .4" between columns; 3.45" text, column 2; .6" edge margin

for a total of 8.5". The text is centered on the page; if a Binding is specified, the text will be displaced in opposite directions on odd and even pages, just as for single-column text. The width of the text in the columns (3.45 in this example) is determined by subtracting all the other space from the 8.5" page width. If there are *nc* columns, the column width is

$$\text{col width} = (8.5" - 2*(\text{edge margin}) - (nc - 1)*(\text{between cols}))/nc$$

The text width and position specified above is only nominal; the actual width and position is determined by the specified left and right margins in the following way. The first column is printed exactly as its left and right margins specify. The second column is moved right by (col width + between cols) from what its left margin specifies (i.e., that amount is added to all its X coordinates). This means, for example, that you can get a double-column page with some text at the top which runs all the way across by setting the right margin of the full-width text appropriately, and using a vertical tab to position the first paragraph of the second column below the full-width text. The appearance of the resulting page will be

Full-width text
first-col text second-col text

Note that to do this you must manually find the end of the first column (easily done using the page boundary command), and put a suitable vertical tab property on the first paragraph of the second column.

A consequence of this laissez-faire approach to column formatting is that you must supply the proper left and right margins yourself. To keep the text within the nominal boundaries defined above, the left margin should be greater than or equal to the edge margin specified in the document profile, and the right margin should be less than or equal to the edge margin plus the column width.

The edge margin specified in the example, which would be much too small for single-column pages, is good for double-column. It is also desirable to reduce the top and bottom margins when you are printing double-column, e.g., to .8" and .4" respectively.

When you are printing more than one column, a L^c in the document starts a new *column* rather than a new page. To start a new page, use two consecutive L^c characters.

LINE NUMBERS

If there is a line which says

Line Numbers: Yes Modulus: *n* Page-relative First Line: *f*

every *n*th line will be numbered, slightly to the left of the standard left margin. Thus, if *n* is 5, the numbers will be 5, 10, 15 If Page-relative appears, numbering starts over on each page; otherwise it continues throughout the document. If First Line appears, the first line is numbered *f*, and numbering continues from there; otherwise the first line is numbered 1.

HEADINGS

If a Heading line appears, it must be followed by a paragraph, also with the profile property, which is used as the heading on each page. This paragraph should have a vertical tab which positions it correctly (for example, 10.5" for the heading on this page) and appropriate margins, centering or whatever to produce the desired effects. It may have more than one line. It is also possible to have separate Odd Heading and Even Heading paragraphs. If Not-on-first-page is present, the heading will not be printed on the first page.

4. Other things

In this section you can learn about a wide variety of other useful things Bravo can do. They are described more-or-less in order of cost-effectiveness: the earlier ones will probably give you more payoff per unit of effort to learn them.

4.1 Some useful features

This section describes a number of features which are easy to learn and helpful to use. As always, it is a good idea to try them out as you read about them.

You can select entire lines of the document by moving the cursor into the *line bar*, which is to the left of the text area and to the right of the scroll bar. You can tell that you are in the line bar because the cursor will appear as a right-pointing arrow when it is in the line bar. To select the entire line pointed to by the cursor, use the RED mouse button. To extend the selection, use the BLUE button. Both of these work very much like selecting a character and extending. The YELLOW button selects a *paragraph*; you can read about paragraphs in section 3.2.

To put the current selection at the top of the screen, say Normalize.

To insert the current *date and time* in front of the current selection, say Time. Notice that it leaves just the time selected, so if you follow it immediately with a Delete, you will be left with just the date. To replace some text with the current date, select it and say Delete Time Delete; be sure you understand why this works.

You can *search* the document for the next occurrence of some text with the Jump command. After you say Jump, you have to specify the text you want to search for, and you do this exactly the way you make an insertion: by typing it in (ending with ESC), by making a copy selection, or by typing ESC to default to the same text which was used for the last Jump (*not* the last insertion or deletion). Notice that if you type in text, it appears between the right-most set of curly brackets in the system window; this is called the *search key buffer*, and it normally contains the last text you searched for. However, the contents of this buffer are destroyed by the Look ? command and some of the Calculator commands. The search starts with the second line displayed in the window. If it succeeds, it brings the first occurrence of the text to the top of the window; if it fails, a note in the system window informs you. Jump does not affect the current selection. The search ignores the looks of the characters.

You can *substitute* one text for another using the Substitute command. It will ask you (in the top window) for the information it needs. In looking for substitutions it will examine only the text in the current selection, so if you want to substitute throughout the document, do an Everything first; this will make the entire document the current selection. For reasons you don't want to know about, it is not a good idea to do a Substitute in which the old text contains a CR.

Most Bravo commands can be *repeated* by simply typing ESC in command mode. When you do this, Bravo uses the *current* selection, not the one which the previous command used. For example, you can append a carriage return after each of several words by selecting the first one and Appending after it, and then selecting successive words and simply typing ESC. Or, you can search through the document looking for occurrences of a word by Jumping to it once and then just typing ESC.

The Undo command will *undo* the action of most Bravo commands which change the document, provided you haven't moved the selection. You can only Undo the most recent command; it will still work if you have scrolled, however.

The `(` command will put parentheses around the current selection. You can put other kinds of brackets around the current selection with the commands `[`, `{`, `<`, `'` and `"`.

The `)` command expands the current selection as little as possible to make it balanced with respect to parentheses. This is useful primarily while editing algebraic expressions or programs. For example, if the current selection is the underlined character in

$$X_{n+1} = ((aX_n) \text{ mod } (m+1) + c) \text{ mod } m$$

then one application of the `)` command will extend the selection thus:

$$X_{n+1} = ((aX_n) \text{ mod } (m+1) + c) \text{ mod } m$$

and another application will do this:

$$X_{n+1} = ((aX_n) \text{ mod } (m+1) + c) \text{ mod } m$$

Again, the `]`, `}`` and `>` commands do similar things.

4.2 Windows

So far you have worked with a single document in a single window. Bravo will let you work on several documents at the same time, each in its own window. This is convenient if you want to compare two documents, or copy something from one into another, say from an address list into a letter. You can also have several subwindows looking into the same document, which is nice when you want to copy something from one part of the document to another, or to check something on another page without losing your place.

At the top of each window, separating it from the one above, is a heavy black bar. Inside this bar is the name of the file for the document in the window; this name is set by Get and used by Put. It can be edited like any other text. Subwindows, created only by the split operation described below, are separated by horizontal black lines. All the subwindows of a window are looking at the same document, although usually at different parts. If part of the document happens to be displayed in several subwindows, any changes to it will appear in all of them, and so will the selection underline or the insertion caret. Two different windows, on the other hand, are always looking at different documents, and no change to one window can affect the other. You can copy text freely from one document to another with a copy selection.

Some commands, like Jump, Everything, Get, Put and Hardcopy, work on the *current* window, which is the window containing the current selection.

There are two commands for windows, one for creating and re-arranging windows, and the other for destroying them. Each has three options, selected by the three mouse buttons.

To *create* a new window, type Window, move the cursor so that it marks the point where you want the new window boundary to be, and hold down BLUE. The new window will appear. As long as you keep BLUE down, you can move the cursor around and the top boundary of the new window will follow it. When you let go of the button, the boundary will freeze. Try it. The new window will be empty, but you can insert or Get into it.

To *split* a window and create a new subwindow, type Window, put the cursor where you want the split, and hold down YELLOW. The new boundary will appear, and it will follow the cursor until you release YELLOW. It is important to understand that after a split you have the *same* document in each subwindow. Scroll one of the subwindows so that some of the same text appears in both subwindows, and select some of the common text. Notice that the selection appears in both subwindows. If you make changes to the document, you will see them in both subwindows. This is very different from creating a new window and Getting the same file into it; that is equivalent to taking another copy out of a file cabinet.

To *move* a window or subwindow boundary, type Window, put the cursor over the boundary you want to move, and hold down RED. The boundary will follow the cursor until you let go of RED.

You can get rid of a window or subwindow by typing Kill, putting the cursor in the doomed window, and holding down RED or BLUE for about a second. RED will give the space of the window to the window above; BLUE will give it to the window below. Kill YELLOW will clear the window; it is equivalent to Everything Delete (except that you can't Undo it).

If you attempt to Kill the only window showing a document that you have changed and haven't filed with Put, Bravo will pause with the message "Type CR to confirm". If you really want to lose the changes you have made to the document, type CR; otherwise, type DEL and then file the document using Put.

In summary:

	<u>Window</u>	<u>Kill</u>
RED	move boundary	give space to window above
YELLOW	split; new subwindow	clear
BLUE	new window	give space to window below

You can remove the *system window* from view by striking the unmarked key to the right of RETURN. This provides more space on the screen and in Alto memory for your document, and also makes subsequent commands execute slightly faster. You can restore the system window by striking the same key again. The system window is restored automatically when you make an error, and at certain other times when Bravo wants you to see what is in the window.

4.3 If Bravo breaks

When Bravo breaks or *crashes*, what usually happens is that Swat gets called; the manifestation is a couple of seconds of whirring from the disk, followed by a mostly blank display on the screen, with the words Swat version xx at the top. If this happens, look at the bottom of the screen, where there will be a more or less intelligible message. In some cases this message may describe a condition you can do something about, e.g., that your disk is full. Or it may inform you of a parity error; if this happens repeatedly, you should file an Alto trouble report to get your Alto repaired (see section 5.1 of the Alto Non-programmer's Guide). A third possibility is some fairly meaningless message describing an internal Bravo malfunction. In any case, after looking at the message you should type Kc (if that doesn't work, boot the Alto). Then, if you want to recover your work, you can proceed as described below.

Bravo makes a record of everything you do during a session; the record is called the *transcript*. It is useful for three reasons:

If Bravo crashes because it has a bug, the transcript can be used to report the problem to the people responsible for fixing bugs.

If Bravo crashes because of a hardware failure of your Alto, a power failure, you accidentally pushing the boot button, or whatever, you can recover your work by *replaying* the transcript. In this case, the last few characters you typed may be lost.

If you make a mistake, like deleting half of the document you have been editing for several hours, you can replay the transcript up to the error and then save the document.

You can do all these things using a system called BravoBug. Thus,

>BravoBugCR

will start replaying your transcript. As the replay proceeds, Bravo will report each command, just as it does when you type a command in the usual way. When it is finished, Bravo will say Ready, and you can go on editing. It is a good idea to save your work with a Put and start Bravo again.

Warning: you can only do a replay if you haven't started Bravo up again. Once you restart Bravo normally, your chance to replay is lost.

To *report* a bug in Bravo and then do a replay, type

>BravoBug/RCR

This will deliver to the Bravo maintainers copies of all the files involved in the bug, including the transcript, and then start Bravo to do the replay.

You can control the replay, step by step, as described in this paragraph. To stop a replay which is going on, type a space. As soon as the command currently being replayed is finished, Bravo will stop and tell you the *number* of the next command. At this point you can type

Quick to make typed-in text go in all at once during the replay, rather than one character at a time. This is faster, but you don't get to see what is going on. Quick is the normal mode.

Slow to make typed-in text go in one character at a time.

space to replay one more command. Note: only commands which change the text or windows are recorded in the transcript, not scrolling operations.

Proceed to continue replaying at full speed. You can stop the replay again at any point by typing a space.

Break before command *n* to make the next Proceed stop before command *n* (of course, it will still stop right away if you type a space). This is useful if you know that the first 50 commands are good, but want to proceed more cautiously after that. Note that after a Bravo crash, the Swat display usually tells you the number of the command during which the crash occurred.

Terminate CR to terminate the replay. After terminating, you can proceed to give ordinary Bravo commands. Don't do this unless you are sure that you want to stop replaying, since there is then no way to resume it.

You should try replaying a Bravo session and using these commands, so that you feel comfortable with them. You will then feel much more in control when you have a problem with Bravo or your Alto, or make a serious blunder while editing.

4.4 Arithmetic

Bravo incorporates a simple calculator, modeled after the Hewlett-Packard 35. The calculator has a *stack* with room for four numbers; while you are using it the top of the stack is displayed in the search key buffer, in the lower right corner of the system window.

To enter a number in your document onto the stack, select it and type \ (enter). To add a number in your document to the top of the stack, select it and type + (or =, which is the lower-case character on the same key). Similarly, you can subtract with -, multiply with * (or ←), and divide with /. The % command is just like *, except that it divides the result by 100. After any of these operations, the top of stack is the current buffer, which means that you can insert its contents in a document by defaulting the text of the insertion with ESC.

If you want to type in a number instead of selecting it, just type the number, and end it with one of the calculator commands. The number will appear in the middle buffer while it is being typed.

You can operate on the top two stack elements, rather than on the current selection and the stack, by prefixing the operation with the Calculator command. Thus, to compute $(a + b) * (c + d)$, you

select <i>a</i>	<u>\</u> (enter)	Stack: <i>a</i>	-	-	-
select <i>b</i>	<u>+</u>	Stack: <i>a+b</i>	-	-	-
select <i>c</i>	<u>\</u>	Stack: <i>c</i>	<i>a+b</i>	-	-
select <i>d</i>	<u>+</u>	Stack: <i>c+d</i>	<i>a+b</i>	-	-
<u>C *</u>		Stack: $(a+b)*(c+d)$	-	-	-

This also works for enter: Calculator \ duplicates the top of the stack; if the stack was *a b c d*, it becomes *a a b c*.

There are a few more calculator commands which are occasionally useful:

~ exchanges the top two elements of the stack: *a b c d* becomes *b a c d*.
↑ rotates the stack, bringing the second element to the top and the top to the bottom: *a b c d* becomes *b c d a*. Four repetitions of ↑ leave the stack where it was.

Calculator n sets the number of digits after the decimal point to *n*. It is initialized to 2. All calculator arithmetic is rounded.

Calculator Fixed sets the display to fixed point (the normal mode).

Calculator Scientific sets the display to scientific notation.

Calculator Engineering sets the display to engineering notation.

Calculator Radix n sets the radix. *n* can be a digit, or Binary, Octal, Decimal or Hexadecimal.

4.5 Other useful features

BUFFERS

The system window contains three pieces of text enclosed in curly brackets. These are called buffers, and they are used for a variety of purposes, some of which you have already encountered. The three buffers are numbered, as follows:

1 {last deletion} 2 {last insertion} 3 {search key}

One of the buffers is always marked with a "*"; that one is the *current* buffer, and its contents are usually what is inserted when you default a text insertion by simply typing ESC.

Commands which insert into buffers, like Jump and Substitute, default to the old contents of the buffer being loaded. Get and Put default to the file name already in the window.

The text in the buffers is always in visible mode, i.e., with spaces, TABS and CRS shown explicitly. Furthermore, TABS and CRS don't have their usual effect of leaving white space, because there is no room in the system window for these effects. Finally, if there is too much text to fit in the space allowed for the buffer on the screen, the middle of the text is replaced by an ellipsis (...).

You can force buffer *n* to be the current buffer with the command Buffer n ESC. You can set the contents of buffer *n* with the command Buffer n followed by typing or a copy selection.

PARTIAL SUBSTITUTION

If you want to substitute "that" for "this" you can use the Substitute command described in section 4.5. If you want to change *some* of the occurrences of "this" to "that", however, it is useful to know about the Find and Yes commands.

Find is exactly like Jump, except that

the search starts from the end of the current selection, not from the top of the display; the occurrence of the key which is found becomes the current selection (Jump leaves the selection unchanged).

Yes is equivalent to Replace ESC Find ESC. So, to change some "this"es to "that"s, proceed as follows:

select something before the first place you want to start looking;

Find this ESC;

ESC to repeat the Find until you get the one you want to change;

Replace that ESC;

Find ESC;

Now at each point type ESC (i.e., repeat the Find) to make no change and go on to the next "this", or Yes to make the same change you made last time and then go on.

CONTROL AND SPECIAL CHARACTERS

Bravo normally displays a control character as the corresponding lower-case letter (or whatever) with an overbar. If you turn on Look graphic it will try to display the control character from the font (if there is something in the font for it; otherwise it will display a black rectangle). This is a character Look, just like Look Visible.

You can't type a control character in directly, but you can type the corresponding regular character, followed by a S^c, which converts the preceding regular character into a control character.

Certain font families (notably TimesRoman and Helvetica) contain extra characters, such as accents, ligatures, and several sizes of dashes and spaces, which are useful in creating very high-quality documents. These may be entered as control characters in graphic mode (use Look graphic). The correspondence between control characters and their printed representations is as follows.

Accents. Each of these has a width of zero so should be typed in immediately *before* the character above which it is to appear.

"	d s ^c	example:	ö
'	e s ^c		ò
'	k s ^c		ó
~	p s ^c		ñ

Ligatures:

ff	f s ^c	replaces:	ff
ffi	q s ^c		ffi
ffi	r s ^c		ffl
fi	t s ^c		fi
fl	u s ^c		fl

Quotation marks:

<code>g s^c</code> <code>,</code> <code>,</code>	open single quote close single quote (standard apostrophe)
--	---

Note that you can make open and close "double quotes" by using pairs of the corresponding 'single quotes'.

Spaces and dashes:

<code> </code> <code> </code> <code> </code> <code>-</code> <code>-</code> <code>-</code>	space bar y s ^c o s ^c - v s ^c x s ^c s s ^c	letter space; here are 10 of them: figure space: em space: hyphen: en dash: minus sign: em dash:	 -----
---	--	--	---

Miscellaneous:

<code>i</code> <code>ç</code> <code>í</code>	b s ^c c s ^c h s ^c
--	--

At the time this manual was prepared, the special characters had not yet been incorporated into the *.al display fonts. Until this has been done, these characters will all appear as black rectangles on the screen. To find out which control character one of these black rectangles corresponds to, turn off its graphic look.

4.6 The user profile and fonts

The file User.cm is your *user profile*, which contains information for various systems about how you want them to be set up for your use. The information for each system starts with the name of the system in brackets, e.g. [BRAVO]. Then follow a series of lines of the form

label: information

Bravo currently accepts three kinds of information in the user profile: initial and quit *macros*, described in the next section, and font definitions and default parameter settings, which are discussed here. Look at your User.cm file now, to see how this works.

Each line of font definition tells

the *number* of the font (0 to 9)

the *name* and *size* (in points) of the hardcopy font

the name and size of one or two screen fonts which can be used to represent that hardcopy font on the screen. Bravo will use the first one normally but the second one (if present) in hardcopy mode.

For example, the line

FONT:0 TIMESROMAN 10 TIMESROMAN 12 TIMESROMAN 10

says that font 0 is to print as 10 pt Times Roman when you generate hardcopy.

The rest of the line says that font 0 can be represented on the screen by the screen fonts stored on files TimesRoman12.al and TimesRoman10.al, which must be present on your disk. Bravo will use the 12 pt version in normal display mode and the 10 pt version in hardcopy mode. The extension ".al" is used for screen fonts. There are no files for the bold or italic versions of screen fonts,

because Bravo can construct them from the regular version.

In addition to all the *.al files, you must also have a file called Fonts.widths, which contains information about the widths of all the characters in the hardcopy fonts.

Warning: A Bravo document file does not contain the association between the font numbers and names used to create that document. If you modify the font definitions in your user profile, you may encounter problems when you attempt to share Bravo documents with other people.

The user profile also contains settings for
default left and right margins;
tab interval;
Look nested parameters;
default line leading (used by Look X) and paragraph leading (used by Look Open, Look Q
and the default for Look Y);
standard offsets (used by Look ↑, Look ←, and the default for Look Up and Look Down).

You can change all these settings by editing User.cm in a way which should be obvious; after doing this, be sure to initialize Bravo. Except for the default margins, all these settings affect only the process of editing the document, and not the document itself. In other words, once a Look nested, superscript or whatever has been done, the margins, offset, etc., have been set in the document and cannot be affected by subsequent changes in the user profile.

There is also a [HARDCOPY] section in the user profile that is used both by Bravo and by other subsystems that generate hardcopy. This contains entries for

preferred printing format (Press is the only reasonable choice at present);
name of your Press printer;
the name to be inserted in the "Printed by" field on the break page of the printout (an occurrence of "\$" is replaced by the name by which you are known to the Alto Executive).

4.7 Startup and Quit macros

This section is only for programmers, and is not recommended even for them.

You can put into your user profile sequences of Bravo commands, called macros, to be executed automatically when you start up or quit from Bravo. Each macro is named by a letter. Startup macro x will be executed if you start Bravo with

>Bravo/x . . . CR

Quit macro x will be executed if you type an "x" instead of CR after typing Quit.

You can see the format of a macro definition by looking at your user profile. The command sequences are just like those which Bravo writes into the transcript (the file Bravo.ts), and can be constructed by actually executing the desired sequence of commands, and then copying Bravo.ts to another file and copying the sequence out of that file. There are two exceptions:

* in place of a selection (which looks like {6,2,123,648}) means the current selection;

@n in place of typed-in text (which looks like 'text') means the *n*th parameter. For startup macros the parameters are strings on the command line separated by blanks; the first one after Bravo/x is numbered 1. For quit macros, parameters 1, 2 and 3 are the three buffers, 4 is the file name for the first window, 5 the file name for the second window, etc.

4.8 Press and Diablo hardcopy

To make a Press file on your disk instead of sending your hardcopy directly to a printer, use the File option of the Hardcopy command. It will ask you for the name of the file; the recommended name is the name of your document, with the extension replaced by "Press". Once you have made this file, you may print it using the EmPress subsystem, store it on a file server to make it available to other people (see sections 6 and 7.4 of the Alto Non-programmer's Guide), or give it to PressEdit to combine it with other files into a large document (see section 7.6 of the Guide). *Warning:* the Press file is typically 50% larger than the document file; be sure you have enough room on your disk.

To send hardcopy to some printer other than your regular one (specified in User.cm), use the Hardcopy option

@ name ESC

where *name* identifies the printer you wish to use.

For both Press and Diablo printing, you can start printing at a specified page number with the Hardcopy option

Start at page *n*

This is mainly useful for Diablo printing. There will usually be a substantial delay while Bravo figures out where page *n* starts.

To print on the Diablo printer, you can use the Diablo option to the Hardcopy command. Before doing this, be sure a printer is plugged into your Alto; when you plug or unplug it, *turn the Alto power off first*. This option has an array of sub-options, which you can invoke when it pauses before printing each page. At the pause, the system window says "Ready to print page *n*", and the beginning of the text for page *n* is displayed in the document window (if there is room in Bravo's memory). You can then say:

Repeat last page to prepare to reprint the previous page, instead of the page which was going to be printed next.

Continuous print, to suppress the pause after each page; this is useful if you have continuous forms in your printer. You can still stop the printing by typing space during printing, as described below.

space to start printing the page which Bravo says it is ready to print.

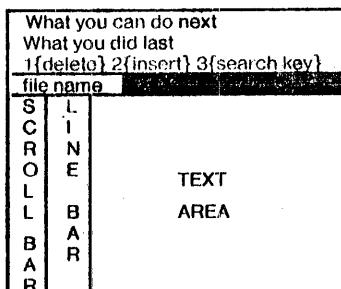
During printing, you can abort printing of the current page at any time by typing a space. You can then use the options just described to restart the current page (with space) or reprint the previous page (with Repeat). If you want to start at another page, use DEL to leave the Hardcopy command, give another Hardcopy command, and use the Start at page *n* option.

There is a command Look Magnify Diablo that displays the document as closely as possible to the way it will appear when printed on the Diablo printer. Look Hardcopy terminates this mode.

BRAVO VERSION 7.4 SUMMARY

Brsum.sil 5-Nov-7

THE SCREEN



SYSTEM
WINDOW

DOCUMENT
WINDOW

THE MOUSE



COMMANDS

	Scroll bar	Line bar	Text area
RED	Scroll up	Select line	Select character
YELLOW	Thumb	Select paragraph (Sets cursor to ¶)	Select word
BLUE	Scroll down	Extend	Extend

The CLR key is the blank key to the right of BS on Alto-I.

or the key labelled BW on Alto-II.

The blank key to the right of RETURN removes or restores the system window.

CHANGING TEXT	Delete selection	(These make a new paragraph if a paragraph is selected.)	During type in:	Backspace character with BS or ctrl-A
	Insert <i>text</i> before selection			Backspace word with ctrl-W
FILING	Append <i>text</i> after selection			Separate paragraphs with ctrl-CR
	Replace selection by <i>text</i>			Enter ctrl- <i>char</i> as <i>char</i> ctrl-S
MISCELLANEOUS	<i>text</i> may be typed or selected. ESC gets the current buffer. Buffer <i>n</i> ESC makes buffer <i>n</i> current.			
	Get to read in a file ctrl-Z for unformatted text		Hardcopy to print. Options: Copies <i>n</i> Start on page <i>n</i> File name .press (no printing) @ printer-name Black (ignore color)	CR to do it Diablo output, with sub-options: Space to start or ab
LOOKS	Put to write out a file			Continuous printing Reprint last page
	File name goes into bar above window. ESC gets you one already there. Can edit file name like any other text.			
Standard Fonts	Undo works on most recent command. ESC repeats most recent command, using current selection. Everything selects whole document. Normalize moves current selection to top of window. ([{ < } put indicated brackets around current selection.)] > extend current selection to closest matching brackets. LF takes current selection as top of page, and moves selection to top of next page.			DEL cancels what you are doing. Quit CR exits from Bravo Time inserts current date and time. Buffer 123 <i>text</i> sets buffer and makes it current. ESC Instead of <i>text</i> just makes it current.
	Basic looks. Mostly, SHIFT means NOT. Thus, Look b turns on bold, Look B turns it off. During type-in, use ctrl-b for bold, etc. In command mode, give the Look command, then the letter alone (no ESC).			
PAGE FORMATTING	Text: bold - underline (highlight if color) italic ←↑ sub/superscript visible 0.9 font graphic CLR to reset		Paragraphs: centered q - add 6 pts nested (indent) ;document profile	Justified open - add 12 pts space before paragraph
	Look has more cases, which take a distance as parameter. These cannot be used during type-in. Text: Up raises text; default 4 pts. Down lowers text; default 4 pts. , <i>n</i> names a tab (<i>n</i> = 1-9, a-e) . c sets color (c = Black, Cyan, Green, Magenta, Red, Violet, Yellow)		Paragraphs: Left margin; default 85 pt F - left margin of first line P - left margin of other lines Right margin; default 527 pt X - space between lines; default 6 pt Y - space between paragraphs; default 12 pt Z - vertical tab; default none Keep on current page; default 0; 11" forces new page TAB n d sets tab stop <i>n</i> (<i>n</i> = 1-9, a-c); default not set TAB = d sets even tab stops, default 36 pt	
WINDOWS	Distances (measured from left or bottom page edge): As a number: in points (72/inch) - 123 or 123 pt. in Inches - 4.5 or 4.5" or 4.5 in in cm - 4.5 cm in blank widths - type that many blanks		As an increment to current value: + or - followed by a number as above As the left edge of a character: select with RED As a position on the screen: point with BLUE Defaulted to a standard value: type \.	
	Look Same <i>selection</i> sets all text looks of current selection to be the same as those of <i>selection</i> Look All <i>selection</i> sets all paragraph looks of current selection to be the same as those of <i>selection</i> Look ? displays looks of selected text or paragraph in buffer 3. Look hardcopy makes screen match hard-copy. Look Hardcopy clears it. Look Magnify Diablo matches Diablo hardcopy			
SEARCHES	The following can be in the document profile; it must be the first paragraph, and have the profile property (Look): Private Data Stamp: Yes/No X:d Y:d Columns:n Edge Margin:d Between Columns:d Margins: Top:d Bottom:d Binding:d Line Numbers: Yes/No Modulus:n Page-relative First Line:n Page Numbers: Yes/No X:d Y:d First Page:n Roman Uppercase Not-on-first-page Heading or Odd/Even Heading: Not-on-first-page followed by a heading paragraph, also with profile property			d is a distance: use - to measure from top or right ctrl-L causes page (or col) break after current line Use two ctrl-Ls for page break with multiple cols Also note Look Keep and Look Z (vertical tab)
	Window RED to move boundary YELLOW to split (make new subwindow) BLUE to make new window Window Edge distance sets the left edge of the window at the specified point on the page; default is 85 pt.		Kill	RED to merge with window above YELLOW to erase contents, leave the window BLUE to merge with window below
REPLAY	Jump <i>text</i> starts search at second line, doesn't move selection Find <i>text</i> starts search after current selection, moves selection to the string found Substitute <i>text</i> for <i>text</i> - works on current selection. Usually you want to do Everything first. Yes is equivalent to Replace ESC Find ESC; use Yes when you want to confirm substitutions.		>Bravobug will start a replay. Space will stop the replay and show you the number of the next command. You can then type: Slow to slow down typein Quick to speed it up	>Bravobug/R will report a bug to the Bravo maintainers first, then replay. Space to replay one more command Proceed to continue replay full speed
				Break n to stop before command n Terminate CR to stop replaying
ARITHMETIC	Operators are + = * * / \% \ (enter). An operator as a command combines the selection with the top of stack. You can also type a number, followed by an operator. The stack is just like the one in a Hewlett-Packard calculator. Also: ~ exchanges x and y ↑ rotates the stack		Calculator, with options: Fixed Scientific Engineering	operator to operate on x and y 0-9 to set digits after decimal pt Radix n to set the radix (<i>n</i> = H for hex)

6

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: ftp.tty

Creation date: 22-Sep-80 11:07:29

For: falis

21 total sheets = 20 pages, 1 copy.

Alto Pup File Transfer Program

FTP is a Pup-based File Transfer Program for moving files to and from an Alto file system. The program comes in 3 parts:

- 1) An FTP Server, which listens for file transfer requests from other hosts.
- 2) An FTP User, which initiates file transfers under control of either the keyboard or the command line, and
- 3) A User Telnet for logging into a remote host using the Pup Telnet protocol.

1. Concepts and Terminology

Transferring a file from one machine (or "host") to another over a network requires the active cooperation of programs on both machines. In a typical scenario for file transfer, a human user (or a program acting on his behalf) invokes a program called an "FTP User" and directs it to establish contact with an "FTP Server" program on another machine. Once contact has been established, the FTP User initiates requests and supplies parameters for the actual transfer of files, which the User and Server proceed to carry out cooperatively. The FTP User and FTP Server roles differ in that the FTP User interacts with the human user (usually through some sort of keyboard interpreter) and takes the initiative in user/server interactions, whereas the FTP Server plays a comparatively passive role.

The question of which machine is the FTP User and which is the FTP Server is completely independent of the direction of file transfer. The two basic file transfer operations are called "Retrieve" and "Store"; the Retrieve operation causes a file to move from Server to User, whereas Store causes a file to move from User to Server.

The Alto FTP subsystem contains both an FTP User and an FTP Server, running as independent processes. Therefore, to transfer files between a pair of Altos, one should start up the FTP subsystem on both machines, then issue commands to the FTP User process on one machine directing it to establish contact with the FTP Server process in the other machine. Subsequent file transfers are controlled entirely from the FTP User end, with no human intervention required at the Server machine.

Transferring files to or from a Maxc system or an IFS involves establishing contact with FTP Server processes that run all the time on those machines. Hence, one may simply invoke the Alto FTP subsystem and direct its FTP User process to connect to the machine.

In the descriptions that follow, the terms "local" and "remote" are

Alto Pup FTP

June 19, 1980

2

relative to the machine on which the FTP User program is active. That is, we speak of typing commands to our "local" FTP User program and directing it to establish contact with an FTP Server on some "remote" machine. A Retrieve command then copies a file from the "remote" file system to the "local" file system, whereas a Store command copies a file from the "local" file system to the "remote" file system.

Furthermore, we refer to "local" and "remote" filenames. These must conform to the conventions used by the "local" and "remote" host computers, which may be dissimilar (for example, Alto versus Maxc). The Alto FTP knows nothing about Maxc filename conventions or vice versa.

The Alto FTP subsystem also includes a third process, called a "User Telnet", which simulates a terminal in a manner exactly analogous to the Chat subsystem (though lacking some of its finer features). By this means, you may log in to a file system machine to perform operations not directly available via the basic file transfer mechanisms. If you log into Maxc, it is even possible to run "PupFTP", the Maxc FTP User program, and direct it to establish contact with the FTP Server in your own Alto. You should probably not try this unless you really understand what you are doing, however, since the terms "local" and "remote" are relative to Maxc rather than to your Alto (since the FTP User program is running on Maxc in this case), which can be confusing.

2. Calling the FTP Subsystem

A number of options are available when running FTP. The program decides which parts of itself to enable and where user commands will come from by inspecting the command line. The general form of the command line to invoke FTP looks like:

```
FTP[/<Global-switches>] [<Host-name> [<Command-list>] ]
```

The square brackets denote portions of the command line that are optional and may be omitted.

Global switches, explained below, select some global program options such as using the Trident disk instead of the Diablo. The first token after the <global-switches>, if present, is assumed to be a <host-name> (a discussion of which appears later in the description of the "Open" command). The User FTP will attempt to connect to the FTP Server on that host. After connecting to the server, if a <command-list> is present, an interpreter is started which feeds these commands to the User FTP. When the command list is exhausted, FTP returns to the Alto Executive. If no command list is present, an interactive keyboard command interpreter is started.

Each global switch has a default value which is used if the switch is not explicitly set. To set a switch to 'false' proceed it with a 'minus' sign (thus FTP/-S means 'no Server'), to set a switch to 'true' just mention the switch.

Switch	Default	Function
--------	---------	----------

Alto Pup FTP

June 19, 1980

3

/S	true	[Server] starts the FTP Server. The Server is not started if the User is enabled and is being controlled from the command line.
/U	true	[User] starts the FTP User. As explained above, the interactive command interpreter or the command line interpreter will be started depending on the contents of the command line.
/C	true	[Chat] starts the Telnet. The Telnet is not started if the User is enabled and is being controlled from the command line, or if the system disk is a Trident.
/T	false	[Trident] sets the system disk to be a Trident drive. The default is 0, but can be changed by following the /T with a unit number. The unit number is octal; the high byte is the logical filesystem number and the low byte is the physical drive number. User and Server commands apply to files on this disk but command line input is still taken from Com.cm on the Diablo drive.
/L	*	[Log] causes all output to the User FTP window to also go to the file "FTP.log" on DP0, overwriting the previous contents. Log is true if the User is enabled and is being controlled from the command line.
/A	false	[AppendLog] enables the log but appends to FTP.log rather than overwriting it.
/E	true	[Error] causes FTP to ask you if you want to continue when a non-fatal error happens during execution of a command line. FTP/-E will cause FTP to recover automatically from non fatal errors without consulting you.
/R	true	[Ram] allows FTP to use some microcode which speeds things up slightly. If your Alto has no ram, this switch is ignored.
/D	false	[Debug] starts FTP in debug mode.

The rest of the global switches are explained below under 'Server Options'.

2.1. FTP User Log

FTP can keep a log (typescript) file for the FTP User window. The file name is 'FTP.log'. It is always enabled when FTP is being controlled from the command line; otherwise it is controlled by the /L and /A global switches.

Alto Pup FTP

June 19, 1980

4

2.2. Using a Trident Disk

Starting FTP with the /T global switch causes FTP to store and retrieve files from a Trident disk. By default, FTP will open TP0; other disks may be opened by appending their unit numbers to the /T switch. Thus "FTP/f1" will open TP1, and "FTP/T400" will open logical filesystem 1 on physical unit 0.

Accessing a file on a Trident requires more code and more free storage than accessing a file on the Diablo. Since FTP is very short on space, only a User or a Server FTP is started when the /T switch is set. The default is to start a User FTP, but specifying no user (FTP/T-U) or specifying a server (FTP/TS) will start a Server FTP instead.

2.3. Server Options

Server options are controlled by switches on the subsystem name and subcommands of the SERVER keyboard command. There are currently four options:

switch	Default	Function
--------	---------	----------

none		If no server option is specified, retrieve requests (disk to net) are allowed. Store requests (net to disk) are allowed unless the store would overwrite an already existing file. Delete and Rename are not permitted.
/P	false	[Protected] Retrieve requests are allowed. No stores are allowed. Delete and Rename are not permitted.
/O	false	[Overwrite] Retrieve requests are allowed. Store requests can overwrite files. Delete and rename are permitted.
/K	false	[Kill] FTP will return to the Alto Exec when the server connection is closed. A simple form of remote job entry can be performed if the user FTP stores into Rem.cm (Com.cm on Novas).

3. The FTP Display

The top inch or so of the display contains a title line and an error window. The title line displays the release date of that version of FTP, the current date and time, the machine's internetwork address, and the number of free pages on the disk. The error window displays certain error messages if they arrive from the network (errors are discussed in more detail below). A window is created below the title line for each part of FTP which is enabled during a session (server, user, and telnet).

If the FTP Server is enabled, it opens a window and identifies itself. If a User FTP subsequently connects to this Server, the User's network address will be displayed. The Server will log the commands it carries

Alto Pup FTP

June 19, 1980

5

out on behalf of the remote User in this window. The Server is not enabled when FTP is being controlled from the command line.

The FTP User opens the next window down and identifies itself. The command herald is an asterisk.

The User Telnet opens the bottommost window, identifies itself, and waits for a host name to be entered. The Telnet is not enabled when FTP is being controlled from the command line.

4. Keyboard Command Syntax

FTP's interactive command interpreter presents a user interface very similar to that of the Alto Executive. Its command structure is also very similar to that of the Maxc Pup FTP program (PupFTP), and the Maxc ArpaNet FTP program (FTP). The standard editing characters, command recognition features, and help facility (via "?") are available. When FTP is waiting for keyboard input, a blinking cursor appears at the next character position.

4.1. Directing Keyboard input to the User and Telnet Windows

The bottom two unmarked keys control which window gets characters from the keyboard. Hitting the unmarked key to the right of 'right-shift' (also known as the 'Swat key') directs keyboard input to the Telnet window. Hitting the unmarked key to the right of the 'return' key (also known as the 'Chat key') directs keyboard input to the FTP User window. The window which currently owns the keyboard will blink a cursor at the next character position if it is waiting for type-in.

4.2. Keyboard Commands

OPEN <host name>

Opens a connection to the FTP Server in the specified host. FTP permits only one user connection at a time. In most cases the word OPEN may be omitted: i.e., a well formed <host name> is a legal command and implies a request to OPEN a connection. FTP will try for one minute to connect to the specified host. If you made a mistake typing the host name and wish to abort the connection attempt, hit the middle unmarked key (to the right of <return>).

Ordinarily, host name should be the name of the machine you wish to connect to (e.g., "Maxc"). Most Altos and Novas have names which are registered in Name Lookup Servers. So long as a name lookup server is available, FTP is able to obtain the information necessary to translate a known host name to an inter-network address.

If the host name of the server machine is not known or if no name lookup servers are available, you may specify an inter-network address in place of the host name. The general form of an inter-network address is:

Alto Pup FTP

June 19, 1980

6

<network> # <host> # <socket>

where each of the three fields is an octal number. The <network> number designates the network to which the Server host is connected (which may be different from the one to which the User host is connected); this (along with the "#" that follows it) may be omitted if the Server and User are known to be connected to the same network. The <host> number designates the Server host's address on that network. The <socket> number designates the actual Server process on that host; ordinarily it should be omitted, since the default is the regular FTP server socket. Hence, to connect to the FTP server running in Alto host number 123 on the directly-connected Ethernet, you should say "OPEN 123#" (the trailing "#" is required).

CLOSE

Closes the currently open User FTP connection. CLOSE cancels any defaults set by CONNECT, DIRECTORY, DEVICE, BYTE, TYPE, or EOLC commands.

LOGIN <user name> <password>

Supplies any login parameters required by the remote server before it will permit file transfers. FTP will use the user name and password in the Operating System, if they are there. Logging into FTP will set the user name and password in the OS (in the same manner as the Alto Executive's "Login" command).

When you issue the "Login" command, FTP will first display the existing user name known to the OS. If you now type a space, FTP will prompt you for a password, whereas if you want to provide a different user name, you should first type that name (which will replace the previous one) followed by a space. The command may be terminated by carriage return after entering the user name to omit entering the password.

The parameters are not immediately checked for legality, but rather are sent to the server for checking when the next file transfer command is issued. If a command is refused by the server because the name or password is incorrect, FTP will prompt you as if you had issued the LOGIN command and then retry the transfer request. Hitting delete in this context will abort the command.

A user name and password must be supplied when transferring files to and from a Maxc system or an IFS. The Alto FTP Server requires a user-password to be supplied if the server machine's disk is password-protected and if the password in the server machine's OS does not match the password on the disk. Thus if the OS was booted and FTP invoked because a Request-for-Connection was received (which bypasses password checking), FTP will refuse access to files unless a password is supplied. However if the OS was booted normally, FTP assumes that the disk owner (who knew the password) will control access by using the server option switches. The user-name is ignored.

CONNECT <directory name> <password>

Requests the FTP server to "connect" you to the specified directory on the remote system, i.e., to give you owner-like access to it. The password may be omitted by typing carriage

Alto Pup FTP

June 19, 1980

7

return after the directory name. As with LOGIN, these parameters are not verified until the next transfer command is issued. CONNECT cancels the effect of any previous DIRECTORY command. At present, the "Connect" command is meaningful only when transferring files to or from a Maxc system or an IFS; the Alto FTP server currently ignores connect requests. If the "multiple directory" feature of the Alto Operating System ever comes into widespread use, this may be changed.

DIRECTORY <directory name>

Causes <directory name> to be used as the default remote directory in data transfer commands (essentially it causes <directory-name> to be attached to all remote filenames that do not explicitly mention a directory). Specifying a default directory in no way modifies your access privileges, whereas CONNECTing gives you 'owner access' (and usually requires a password). Explicitly mentioning a directory in a file name overrides the default directory, which overrides the connected directory, which overrides the login directory. Punctuation separating <directory name> from other parts of a remote filename should not be included. For example you might type "Directory Alto" not "Directory <Alto>".

RETRIEVE <remote filename>

Initiates transfer of the specified remote file to the local host. The syntax of <remote filename> must conform to the remote host's file system name conventions. Before transferring a file, FTP will suggest a local-filename (generally the same as the remote-filename without directory or version), and will tell you whether or not the file already exists on your local disk. At this point you may make one of three choices:

1. Type Carriage Return to cause the data to be transferred to the local filename.
2. Type Delete to indicate that the file is not to be transferred.
3. Type any desired local filename followed by Return. The previous local filename will disappear, the new filename will replace it, and FTP will tell you whether a file exists with that name. This filename must conform to local conventions. You now have the same three choices.

If the remote-filename designates multiple files (the remote host permits "*" or some equivalent in file names), each file will be transferred separately and FTP will ask you to make one of the above three choices for each file. At present, only Maxc and IFS support this capability. That is, you may supply "*"s in the remote-filename when retrieving files from a Maxc or an IFS, but not when retrieving files from another Alto.

STORE <local filename>

Initiates transfer of the specified local file to the remote host. Alto file name conventions apply to the <local filename>; "*" expansion is not supported. FTP will suggest a remote-filename to which you should respond in a manner similar to that described under RETRIEVE except that if you supply a different filename, it must conform to the remote file system's conventions. The default

Alto Pup FTP

June 19, 1980

8

remote filename is one with the same name and extension as the local file; the remote server defaults other fields as necessary. If the remote host is a Maxc system or an IFS, then the directory is that most recently supplied in LOGIN or CONNECT or DIRECTORY commands and the version is the next higher.

DUMP <remote filename>

Bundles together a group of files from the local file system into a 'dump-format' file (see the Alto Executive documentation for the dump-file format and more on dump-files in general) and stores the result as <remote filename>. FTP will ask you for the names of local files to include in the dump-file. Terminate the dump by typing just <return> when FTP asks for another filename. By convention, files in dump-format have extension '.dm'.

LOAD <remote filename>

Performs the inverse operation of DUMP, unbundling a dump-format file from the remote file system and storing the constituent files in the local file system. For each file in the dump-file, FTP will suggest a local file name and tell you whether a file by that name exists on your disk. You should respond in the manner described under RETRIEVE.

LIST <remote file designator>

Lists all files in the remote file system which correspond to <remote file designator>. The remote file designator must conform to file naming conventions on the remote host, and may designate multiple files if "*" expansion or some equivalent is supported there. If the <remote file designator> is terminated by <comma carriage return> rather than just a <carriage-return>, FTP prints a prompt of "##" at the left margin and prepares to accept one or more subcommands. These subcommands request printout of additional information about each file. To terminate subcommand input, type a <return> in response to the subcommand prompt. The subcommands are:

Type	Print file type and byte size.
Length	Print length of file in bytes.
Creation	Print date of creation.
Write	Print date of last write.
Read	Print date of last read.
Times	Print times as well as dates.
Author	Print author (creator) of file.
Verbose	Same as Type+Write+Read+Author.
Everything	Print all information about the file.

This information is only as reliable as the Server that provided it, and not all Servers provide all of these file properties. Altos derive much of this information from hints, so do not be alarmed if it is sometimes wrong.

DELETE <remote filename>

Deletes <remote filename> from the remote filesystem. The syntax of the remote filename must conform to the remote host's file system name conventions. After determining that the remote file exists, FTP asks you to confirm your intention to delete it. If the remote filename designates multiple files (the remote host permits "*" or some equivalent in file names), FTP asks you to confirm the deletion of each file.

Alto Pup FTP

June 19, 1980

9

RENAME <old filename> <new filename>
Renames <old filename> in the remote filesystem to be <new filename>. The syntax of the two filenames must conform to the remote host's file system name conventions, and each filename must specify exactly one file.

QUIT

Returns control to the Alto Executive, closing all open connections.

TYPE <data type>

Forces the data to be interpreted according to the specified <data type>, which may be TEXT, BINARY. Initially the type is UNSPECIFIED, meaning that the source process should, if possible, decide on the appropriate type based on local information.

BYTE-SIZE <decimal number>

Applicable only to files of type Binary, BYTE-SIZE specifies the logical byte size of the data to be transferred. The default is 8.

EOL <convention>

Applicable only to files of type Text, EOL specifies the End-of-line Convention to be used for transferring text files. The values for <convention> are CR, CRLF, and TRANSPARENT. The default is CR.

DEVICE <device>

Causes <device> to be used as the default device in data transfer commands (essentially it causes <device> to be attached to all remote filenames that do not explicitly mention one). The punctuation separating <device> from the other components of a remote filename should not be included. For example you might specify "Device DSK" to Tenex, not "Device DSK:"

USER

Allows you to toggle switches which control operation of the FTP User. There is currently only one: DEBUG, which controls display of protocol interactions. Warning: this printout (and the corresponding one in the SERVER command below) sometimes includes passwords.

SERVER

Allows you to toggle switches which control operation of the FTP Server. The switches are PROTECTED, OVERWRITE, KILL, and DEBUG, corresponding to the global switches /P, /O, /K, and /D.

TELNET

Allows you to toggle switches which control operation of the Telnet. There is currently only one: CLOSE, which closes the Telnet connection if one is open, and clears the Telnet window.

6. Command Line Syntax

The User FTP can also be controlled from the command line. As explained above, the first token after the subsystem name and server

Alto Pup FTP

June 19, 1980

10

switches must be a legal host name; if the User FTP can't connect to the FTP Server on that host it will abort and return control to the Alto Executive. If a command list follows the host name, the command line interpreter is invoked instead of the interactive keyboard interpreter. This permits the full capabilities of the Alto Executive (filename recognition, "*" expansion, command files, etc.) to be used in constructing commands for FTP.

Each command is of the form:

<Keyword>/<SwitchList> <arg> ... <arg>

To get a special character (any one of "#';") past the Alto Executive, it must be preceded by a single quote. To get a "/" into an FTP argument, the "/" must be proceeded by two single quotes (the second one tells FTP to treat the "/" as an ordinary character in the argument, and the first one gets the second one past the Alto Executive).

Unambiguous abbreviations of command keywords (which in most cases amount to the first letter) are legal. However, when constructing command files, you should always spell commands in full, since the uniqueness of abbreviations in the present version of FTP is not guaranteed in future versions.

A command is distinguished from arguments to the previous command by having a switch on it, so every command must have at least one switch. The switch "/C" has no special meaning and should be used on commands where no other switches are needed or desired.

5.1. Command Line Errors

Command line errors fall into three groups: syntax errors, file errors, and connection errors. FTP can recover from some of these, though it leaves the decision about whether to try up to you.

Syntax errors such as unrecognized commands or the wrong number of arguments to a command cause FTP's command interpreter to get out of sync with the command file. FTP can recover from syntax errors by simply ignoring text until it encounters another command (i.e. another token with a switch).

File errors such as trying to retrieve a file which does not exist are relatively harmless. FTP recovers from file errors by skipping the offending file.

Connection errors such as executing a store command when there is no open connection could cause FTP to crash. FTP can't recover from connection errors.

When FTP detects an error, it displays an error message in the User window. If the error is fatal, FTP waits for you to type any character and then aborts, causing the Alto Executive to flush the rest of the command line, including any commands to invoke other subsystems after FTP. If FTP can recover from the error, it asks you to confirm whether you wish to continue. If you confirm, it plunges on, otherwise it aborts. The confirmation request can be bypassed by invoking FTP with the global error switch false (FTP/-E ...) in which case it will plunge

Alto Pup FTP

June 19, 1980

11

on after all non fatal errors. If you aren't around when an error happens and you have told FTP to get confirmation before continuing after an error, the remote Server will probably time out and close the connection. If you then return and tell FTP to continue, it will get a fatal connection error and abort.

6.2. Command Line Commands

OPEN/C <host name>

See description in "Keyboard commands". The first token after the subsystem name and global switches is assumed to be a host name and no OPEN verb is required (in fact if you supply it, FTP will try to make a connection the host named OPEN which is almost certainly not what you want).

CLOSE/C

Closes the currently open User FTP connection.

LOGIN/C <user name> <password>

See description in "Keyboard commands". The <password> may be omitted.

LOGIN/Q <user name>

Causes FTP to prompt you for the password. This form of LOGIN should be used in command files since including passwords in command files is a bad practice.

CONNECT/C <directory name> <password>

See description in "Keyboard commands". The <password> may be omitted.

CONNECT/Q <directory name>

Causes FTP to prompt you for the password needed to connect to the specified <directory name>. This form of CONNECT should be used in command files since including passwords in command files is a bad practice.

DIRECTORY/C <default directory>

See description in "Keyboard commands".

RETRIEVE/C <remote filename> ... <remote filename>

Retrieves each <remote filename>, constructing a local file name from the actual remote file name as received from the Server. FTP will overwrite an existing file unless the /N (No overwrite) switch is appended to the RETRIEVE command keyword. If the remote host allows "*" (or some equivalent) in a filename, a single remote filename may result in the retrieval of several files. (Note that you must quote the "*" to get it past the Alto Executive's command scanner.) As mentioned previously, this capability is implemented only by Maxc and IFS FTP Servers at present.

RETRIEVE/S <remote filename> <local filename>

Retrieves <remote filename> and names it <local filename> in the local file system. This version of RETRIEVE must have exactly two arguments. FTP will overwrite an existing file unless the /N (No overwrite) switch is also appended to the RETRIEVE command keyword. The remote filename should not cause the server to send multiple files.

Alto Pup FTP

June 19, 1980

12

RETRIEVE/U <remote filename> ... <remote filename>

Retrieves <remote filename> if its creation date is later than the creation date of the local file. A file will not be retrieved unless a local file with name and extension equal to the name and extension of the remote filename exists, or if the FTP server does not send a CREATION-DATE property. This option can be combined with RETRIEVE/S to rename the file as it is transferred.

RETRIEVE/V

Requests confirmation from the keyboard before writing a local file. This option is useful in combination with the Update option since creation date is not a fool-proof criterion for updating a file.

STORE/C <local filename> ... <local filename>

Stores each <local filename> on the remote host, constructing a remote filename from the name body of the local filename. A local filename may contain "*", since it will be expanded by the Alto Executive into the actual list of filenames before the FTP subsystem is invoked.

STORE/S <local filename> <remote filename>

Stores <local filename> on the remote host as <remote filename>. The remote filename must conform to the file name conventions of the remote host. This version of store must have exactly two arguments.

DUMP/C <remote filename> <local filename>...<local filename>

See the description in "Keyboard Commands".

LOAD/C <remote filename> ... <remote filename>

See the description in "Keyboard Commands". If the /V switch is appended to the LOAD command keyword, FTP will request confirmation before writing each file. Type <return> to write the file, to skip it. FTP will overwrite an existing file unless the /N (No overwrite) switch is appended to the LOAD command keyword.

LOAD/U <remote filename> ... <remote filename>

Loads files from <remote filename> if their creation dates are later than the corresponding creation dates of local files. A file will not be retrieved unless a local file with name and extension equal to the name and extension of the remote filename exists, or if no creation date is included in the dump file.

LOAD/S <remote filename> <filename 1> ... <filename n>

Loads files from <remote filename> if their names are in the list <filename 1> ... <filename n>. Files within the dump file that are not in the list are skipped. This option can be combined with the /U, /V, and /N options.

LIST/C <remote filename> ... <remote filename>

See the description in "Keyboard Commands". The subcommands are specified by local switches: /T = type, /L = length in bytes, /D = creation date, /W = write date, /R = read date, /A = author, /V = verbose, /E = everything. Dates always include times; /C should have been the creation date but that collides with the use of /C to mean no local options (sigh).

DELETE/C <remote filename>

Alto Pup FTP

June 19, 1980

13

See the description in "Keyboard Commands". If the /V switch is appended to the DELETE command keyword, FTP will request confirmation before deleting each file. Type <return> to delete the file, and (oops!) if you don't want to delete it.

COMPARE/C <remote filename>...<remote filename>

Compares the contents of <remote filename> with the file by the same name in the local file system. It tells you how long the files are if they are identical or the byte position of the first mismatch if they are not. (No corresponding command is available in the Keyboard command interpreter for implementation reasons: there is not enough room for it in Alto memory.)

COMPARE/S <remote filename> <local filename>

Compares <remote filename> with <local filename>. The remote filename must conform to the file name conventions of the remote host. This version of COMPARE must have exactly two arguments.

RENAME/C <old filename> <new filename>

See the description in "Keyboard Commands".

TYPE/C <data type>

See the description in "Keyboard Commands".

BYTE-SIZE/C <decimal number>

See the description in "Keyboard Commands".

EOL/C <convention>

See the description in "Keyboard Commands".

DEVICE/C

See the description in "Keyboard Commands".

DEBUG/C

See the description of the DEBUG subcommand under the USER command in "Keyboard Commands".

5.3. CLI Examples

To transfer files FTP.run and FTP.syms from the Alto called "Michelson" to the Alto called "Morley", one might start up FTP on Michelson (to act as an FTP Server), then walk over to Morley and type:

FTP Michelson Retrieve/c FTP.run FTP.syms

Alternatively, one could start an FTP server on Morley (invoking it by "FTP/O" to permit files to be overwritten on Morley's disk), then issue the following command to Michelson:

FTP Morley Store/c FTP.run FTP.syms

The latter approach is recommended for transferring large groups of files such as "*.run" (since expansion of the "*" will be performed by the Alto Executive).

To retrieve User.cm from the FTP server running on Alto serial number 123 (name unknown, but it is on the local Ethernet):

FTP 123'# Retrieve User.cm

Alto Pup FTP

June 19, 1980

14

Note that the "#" must be preceded by a single quote when included in a command line, since otherwise the Alto Executive does funny things with it. (Quotes are not necessary when typing to FTP's interactive keyboard interpreter).

To start FTP, have the FTP User connect to Maxc, and then accept further commands from the keyboard:

FTP Maxc

To retrieve <System>Pup-Network.txt from Maxc and store it on the Alto as PupDirectory.bravo, and store PupRTP.bcpl, Pup1b.bcpl, and PupBSPStreams.bcpl on <DRB> with their names unchanged:

FTP Maxc Connect/c drb mypassword Retrieve/s <System>Pup-Network.txt PupDirectory.bravo Store/c PupRTP.bcpl Pup1b.bcpl PupBSPStreams.bcpl

To retrieve the latest copy of all .RUN files from the <alto> directory, overwriting copies on the Alto disk (The single quote is necessary to prevent the Alto Executive from expanding the "*"):

FTP Maxc Ret/c <alto>'*.run

To update the Alto disk with new copies of all <alto> files whose names are contained in file UpdateFiles.cm, requesting confirmation before each retrieval:

FTP Maxc Dir/c Alto Ret/u/v @UpdateFiles.cm@

To store all files with extension .BCPL from the local Alto disk to your login directory on Maxc (the Alto Executive will expand "*.bcpl" before invoking FTP):

FTP Maxc St/c *.bcpl

To retrieve <System>Host-name/descriptor-file.txt';43 (two single quotes are necessary to get the "/" past the Alto Executive and the FTP command scanner, and one quote is necessary to get the ";" past the Alto Executive):

FTP Maxc Ret/c <System>Host-name''/descriptor-file.txt';43

To send Prog.f4, Data.f4, and Command.f4 to Fortran-Machine and then cause the FTP server on Fortran-Machine to quit (presumably to execute Prog.f4 on Data.f4 according to the commands in Command.f4):

FTP Fortran-Machine Store/c Prog.f4 Data.f4 Store/s
Command.f4 Rem.cm

FTP on Fortan-Machine must be started with the /K server option switch, and Command.f4 should re-invoke FTP as its last act so that the results can be retrieved.

To release a new version of FTP, I incant:

@ReleaseAltoFTP.cm@

which the Alto Executive expands into:

Alto Pup FTP

June 19, 1980

15

```
FTP Maxc Connect/q Alto Store/c FTP.run FTP.syms Connect/q
AltoSource Dump/c FTP.dm @ftp.cm@
```

and then into:

```
FTP Maxc Connect/q Alto Store/c FTP.run FTP.syms Connect/q
AltoSource Dump/c FTP.dm @FtpSubsys.cm@ @FtpPackage.cm@
FTP.cm
```

and finally into:

```
FTP.run Maxc Connect/q Alto Store/c FTP.run FTP.syms
Connect/q AltoSource Dump/c Ftp.dm Ftp.bcp1 FtpNv.bcp1
FtpInit.bcp1 FtpInit1.bcp1 FtpNvInit.bcp1 FtpUserInit.bcp1
FtpSubsys.decl FtpKbdInit.bcp1 FtpKbd.bcp1 FtpKbd1.bcp1
FtpKbd2.bcp1 FtpCliInit.bcp1 FtpCli.bcp1 FtpCli1.bcp1
FtpCli2.bcp1 FtpCliUtil.bcp1 FtpMiscb.bcp1 FtpMisca.asm
FtpServerInit.bcp1 FtpServer.bcp1 FtpTelnetInit.bcp1
FtpTelnet.bcp1 FtpKeys.bcp1 FtpCmdScanDsp.bcp1 FtpMc.mu
FtpRamTrap.mu CompileFtpmc.cm FtpSubsys.cm
CompileFtpSubsys.cm CompileAltoFtp.cm LoadAltoFtp.cm
MakeHiddenFtp.cm LoadHiddenFtp.cm ReleaseAltoFtp.cm
CompileNovaFtp.cm LoadDosFtp.cm LoadRDosFtp.cm FtpProt.decl
FtpUserProt.bcp1 FtpUserProtFile.bcp1 FtpUserProtMail.bcp1
FtpServProtFile.bcp1 FtpServProtMail.bcp1 FtpPLListInit.bcp1
FtpPLListProt.bcp1 FtpPLList1.bcp1 FtpUtilInit.bcp1
FtpUtil1B.bcp1 FtpUtilA.asm FtpUtilXfer.bcp1 FtpUtilDmpLd.bcp1
FtpUtil1CompB.bcp1 FtpUtil1CompA.asm BlockEq.mu FtpOEPIInit.bcp1
CompileFtpPackage.cm DumpFtpPackage.cm FtpPackage.cm Ftp.cm
```

To load Ftp.dm from <AltoSource>, expanding it out into its constituent files:

```
FTP Maxc Load/c <AltoSource>Ftp.dm
```

To cause Memo.ears to be spooled for printing on Ears by the Maxc printing system:

```
FTP Maxc Store/s Memo.ears LPT:
```

This also works for Press files and unformatted text files if you know what you are doing. It does not do the right thing for Bravo-format files.

To use FTP as a stop-gap IFS:

```
FTP/T-U0
```

This starts only a server with overwriting of existing files permitted. When using the trident, there isn't enough space to start both a User and a Server.

6. File Property Defaulting

Without explicit information from the file system, it is often difficult to determine whether a file is Binary or Text, if Binary,

Alto Pup FTP

June 19, 1980

16

what its byte-size is, and if Text, what End-Of-Line convention is used. The User and Server FTPs use some simple heuristics to determine the correct manner in which to transfer a file. The heuristics generally do the right thing in the face of incomplete information, and can be overridden by explicit commands from a human user who knows better.

The FTP protocol specifies a standard representation for a file while in transit over a network. If the file is of type Binary, each logical byte is packed right-justified in an integral number of 8-bit bytes. The byte-size is sent as a property along with the file. If the file is of type Text, each character is sent right-justified in an 8-bit byte. An EOL convention may be sent as a file property. The default is that <return> marks the end of a line.

6.1. File Types

FTP determines the type of a local file by reading it and looking for bytes with the high-order bit on. If any byte in the file has a high-order bit on, the file is assumed to be Type Binary, otherwise it is assumed to be Type Text. FTP will generate a warning, but allow you to send what it thinks to be a text file as type Binary, since no information is lost. It will refuse to send a binary file as type text.

Don't specify a Type unless you know what you are doing. The heuristic will not lose information.

6.2. Byte-Size

If a file is type Binary, the byte-size is assumed to be 8 unless otherwise specified. The FTP User and Server will both accept binary files of any byte-size and write them as 8 bit bytes on the disk. No transformation is done on the data as it is written to the disk: it is stored in network default format. Since there is no place in the Alto file system to save the byte-size property, it is lost.

Similarly, requests for Binary files will be honored with any byte size, and whatever is on the disk will be sent to the net without transformation. Since Alto files have no byte size information, the byte-size property will be defaulted to 8 unless otherwise specified (by the BYTE command), in which case whatever was otherwise specified will be sent as the byte size.

Don't specify a Byte-size unless you know what you are doing. Alto-Alto transfers can't go wrong. Alto-Maxc transfers with weird byte-sizes will not work unless the byte-size specified in the Alto to Maxc direction is the same as the byte-size in which the file was stored on the Alto. If it isn't, the Alto will not give any error indication, but the result will be garbage.

6.3. End-of-Line Conventions

FTPs are expected to be able to convert text files between the local file system End-Of-Line (EOL) convention and the network convention.

Alto Pup FTP

June 19, 1980

17

Conveniently enough, the Alto file system's internal representation of a text file is the same as the network standard (a bare <return> marks the end of a line). The Alto FTP does not do any transformations on text files. It will refuse to store a text file coming in from the net whose EOL convention is CRLF.

As an escape to bypass conversion and checking, EOL convention 'transparent' tells both ends NOT to convert to network standard, but rather send a file 'as is'. This is included for Lisp files which contain internal character pointers that are messed up by removing line feed characters.

Don't specify an EOL convention unless you know what you are doing. If your text file is a Lisp source file, specify EOL convention 'Transparent'.

6.4. File Dates

The Alto file system keeps three dates with each file: Creation, Read, and Write. FTP treats the read and write dates as properties describing the local copy of a file: when the file was last read and written in the local file system. FTP treats the creation date as a property of the file contents: when the file contents were originally created, not when the local copy was created. Thus when FTP makes a file on the local disk, the creation date is set to the creation date supplied by the remote FTP, the write date is set to 'now' and the read date is set to 'never read'.

7. Abort and Error messages

Error and Abort packets are displayed in a window above the title line. Abort packets are fatal; Error packets are not necessarily so.

The most common Abort message is "Timeout. Good bye", generated when a server process has not received any commands for a long time (typically 5 minutes).

The most common Error message is "Port IQ overflow" indicating a momentary shortage of input buffers at the remote host. Receiving an Error Pup does not imply that the file in transit has been damaged. Loss of or damage to a file will be indicated by an explicit message in the User FTP window. The next iteration of Pup will probably rename 'Error Pups' to be 'Information Pups'.

8. Telnet

FTP provides a simple User Telnet as a convenience for logging into a remote host (e.g., Maxc) to poke around without having to leave the FTP subsystem and start Chat. It lacks most of the creature comforts Chat provides, such as automatic attaching to detached jobs, automatic logging in, etc. The Telnet is not enabled when the User FTP is being controlled from the command-line. When the Telnet does not have an

Alto Pup FTP

June 19, 1980

18

open connection, it waits for you to type a host name with the syntax explained above for the OPEN command, and then attempts to connect to the specified host. If you wish to abort the connection attempt, hit the bottom unmarked key (opposite right-shift). You can get a larger Telnet window by not starting a server (type FTP/-S to the Executive).

9. Nova FTP

FTP is also available running under Dos Rev 4 and RDos Rev 3. Since the Nova versions are nearly identical to the Alto version (the same source files except for initialization), only the differences are listed here.

- 1) Ignore all references to display windows. All printout goes to device #11, whatever that is.
- 2) Ignore all references to 'unmarked keys' such as for aborting connection attempts and directing keyboard input to various windows.
- 3) Lack of memory and lack of a windowing display made including a Telnet impractical on the Nova.
- 4) The syntax of the command line is limited to that acceptable to the Nova operating system. Warning: the command line examples given above may not all work on a Nova.
- 5) The Nova OS does not maintain a username or password, so all interactions with a Maxc system or an IFS will require the user to supply them.
- 6) File creation dates are not supported, so there is no Update option to RETRIEVE, and the LIST command does not show dates.

9.1. FTP releases

The Nova FTP subsystem consists of a save-file, FTP.SV, and an overlay-file, FTP.BB. You must get BOTH files when a new version of FTP is released. If you rename FTP.SV you must rename FTP.BB to have the same name (for instance if you rename FTP.SV to be OLDFTP.SV you must also rename FTP.BB to be OLDFTP.BB). New releases of FTP will be distributed as dump files with a consistant pair of save- and overlay-files.

9.2. Device codes

FTP assumes that Nova Ethernet interfaces have device codes 73 and 74, 63 and 64, or 53 and 54. It will use all interfaces with these codes that seem (from reading some status registers) to be Ethernets. The Dos version of FTP assumes that Nova MCA interfaces are device code 6 and 7, or 46 and 47. It will use all interfaces with these codes that seem to be MCAs.

Alto Pup FTP

June 19, 1980

19

9.3. RDos notes

FTP is big, and will not run under some RDos systems. If you have trouble, generate a smaller system and boot from it when running FTP. FTP disables parts of RDos with patches which may not work for versions other than Rev 3. It will NOT work under an RDos that uses the memory map hardware. The RDos version does not include MCA drivers.

10. Revision History**April 1976**

First release.

May 1976

/Q switch added to CONNECT. Connection requests to the User FTP and Telnet can be aborted. Login prompt changed. 1 minute Timeout added when waiting to finish after a command line error. User FTP automatically recovers from more "No" responses from the remote server.

June 1976

Dos version released. DIRECTORY and LIST commands added. Update (/U) option added. File creation dates added. 5 minute no-activity timeout added to FTP Server. FTP version, time-of-day, and machine address added in top window. "Ding" now flashes only the affected window instead of the whole display.

August 1976

RDos version released. Same as June release for Dos and Alto.

October 1976

DUMP and LOAD commands added to user FTP. KILL command added. Free disk page count added to the title line. Verify (/V) switch added to the RETRIEVE command.

November 1976

Bug fixes to the October release.

May 1977

This version was only released to friends. KILL command removed and turned into a server option. DEBUG command moved into new USER and SERVER commands. Trident disk option (/T) added. User LIST command improved and Server LIST response implemented. Password checking by the FTP server implemented. Telnet window enlarged at the expense of possibly losing information from the top of the window if the lines are very full. DELETE, RENAME, and DEVICE commands implemented. Much internal reorganization so that the protocol modules could be used in IFS and released as a package.

July 1977

Alto Pup FTP

June 19, 1980

20

Global switches changed. <Shift-Swat> should work more reliably now. User LIST command further improved. Keyboard command interpreter is much more robust and consistant. Command line STORE and DUMP go much faster since they look up files using MDI. FTP/Tx opens Trident unit 'x'. LOGIN command added to command line interpreter.

November 1977

Special microcode added to speed up execution.

March 1978

User log option added (see /L and /A switches and 'FTP User Log' section). AllocatorDebug switch removed. New command line commands COMPARE, OPEN, and CLOSE added. Command line errors are handled differently (see /E global switch and 'Command Line Errors' section). When using a Trident, either a User or a Server FTP is started but not both (see the section on Trident disks).

September 1979

This is a maintenance release coordinated with OS17, fixing a few bugs and reloading with current packages. CONNECT cancels any previous DIRECTORY. CLOSE cancels any previous CONNECT, DIRECTORY, DEVICE, TYPE, BYTE, or EOLC. Multiple logical file systems on a T-300 can now be addressed: Ftp/T400 opens logical filesystem 1 on physical unit 0.

October 1979

The command line version of the OPEN command retries failed connection attempts every five seconds under control of the error flag. Ftp.boot is now a type B boot file. It EtherBoots faster and consumes less disk space in boot servers. It now works with all Alto file system configurations.

June 1980

New command line commands: LIST, LOAD/U and LOAD/S. Ftp handles file creation dates in dump-format files. Subcommand mode in the keyboard LIST command is the same as Maxc and IFS, namely one terminates the filename with <comma carriage-return>, and the VERBOSE option includes file lengths. The keyboard DIRECTORY and DEVICE commands display their previous values. Data transfer commands report bytes transferred and bits per second. The TFS option now works on Alto/Sugart systems.

7

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: laurelmanual.press

Creation date: 14-Dec-79 11:49:24 PST

For: falis

22 total sheets = 21 pages, 1 copy.

Problems encountered:

Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.
Not impl.: brightness, hue, saturation, show-object, show-dots-opaque, dots from files.

Laurel Manual

Table of Contents

1. Introduction	65
What is Laurel?	
What is this manual?	
2. Basic notions and facilities	65
How to obtain Laurel	
How to run Laurel	
2.1 The user interface	66
Command invocation	
Type-in conventions	
Brackets	
2.2 The display	67
Scrolling and thumbing	
Adjusting region sizes	
2.3 The table-of-contents region	68
2.4 The upper menu	69
Mail files	
New mail	
2.5 The selection commands menu	70
Displaying messages	
Selecting messages	
Deleting and undeleting messages	
Generating hardcopy	
2.6 Composing and delivering messages	71
Composition	
Delivery	
2.7 The feedback region	72
2.8 Leaving Laurel	73
3. Additional facilities	74
3.1 Filing and classifying messages	74
Move to {file}	
Mark characters	
3.2 Additional editing and delivery facilities	74
Secondary selection	
Paragraphs	
Distribution lists	
Get and Put	
Message header format	
3.3 Authentication and logging in	77
3.4 Polling for new mail	77
3.5 Command line options	77
Mail file selection	
In-box interrogation	
Send message mode	

3.6 The Laurel profile	78
4. If things go wrong . . .	80
5. Laurel and MSG	80
Mail files	
Mail file philosophy	
Overwrite	
Processing your mail away from an Alto	
Moving mail files from Maxc	
6. Look before you leap . . .	82
Formatting messages for non-Laurel users	
Secondary selection from the message display region	
Tabs	
The Answer form	
File version numbers	
Space required for mail files	
Hardcopy usage	
7. Things a casual user doesn't really need to know	83

1. Introduction

What is Laurel?

Laurel is an Alto-based, display-oriented, message manipulation system. It provides facilities that permit its users to display, forward, classify, file, and print messages, and to compose and transmit messages and replies. Laurel is an initial component of what will ultimately be a distributed message system. Although the distributed nature of this system has inherent technical interest, it is largely transparent to the users of the system, who see a collection of logical facilities resembling those provided by MSG on Maxc. Eventually, the services of Laurel will surpass those of MSG, but at present, the two are roughly equivalent in function. The important distinction for now is that Laurel executes on an Alto and uses the display in a fashion befitting Alto-based software. It also produces files that can be manipulated by other Alto subsystems.

Many initial users of Laurel will be familiar with MSG and will naturally be interested in the functional differences between MSG and Laurel. This manual, in addition to presenting the facilities of Laurel, points out some of the incompatibilities with MSG. *Prospective Laurel users with strong ties to MSG should read section 5 carefully before committing themselves to Laurel.* The Laurel team fully expects that some potential users will find the transition to Laurel too uncomfortable to undertake at present. Accordingly, we have provided a "free sample" of Laurel in the form of a tutorial (see section 2), which can be run without any commitment to further use of Laurel or danger to MSG files. We would much prefer that Laurel users "go in with their eyes open" and not be unpleasantly surprised when confronted by incompatibilities.

What is this manual?

This manual is a reference document for Laurel. The Laurel team believes that most of the basic facilities of Laurel are self-explanatory, and that you can probably use Laurel quite competently after reading only the introductory sections of this manual. (There is also a tutorial available to guide new users; see section 2.) However, Laurel has features that are not immediately obvious, and after becoming acquainted with the system, you will want to read about these facilities. The Laurel team will be very happy to hear any suggestions you may have, and is particularly interested in your experiences in using the initial system. Comments should be sent to LaurelSupport (using the facilities provided by Laurel!)

The version described in this manual is Laurel 2.0. Laurel displays its version number in the upper-left corner of the screen. Future versions of Laurel containing major new capabilities will change the integer part of the version number, whereas maintenance releases will change the fractional part.

2. Basic notions and facilities

Laurel is a display-based, interactive program that manipulates a particular class of files, called *mail files*. In essence, a mail file is just a sequence of *messages*, each of which is a text string formatted according to certain conventions. The details of these conventions are not of major interest to most users—suffice it to say that messages have a *header* and a *body*; a header contains (at least) a *sender*, a *subject*, one or more *recipients*, and a *date*. For each mail file, Laurel constructs and maintains a *table-of-contents* that summarizes the messages residing in the file. Mail files are private to each user, meaning, at present, that your mail files reside on your own Alto disk(s). Laurel provides facilities for manipulating mail files, examining and responding to messages, composing and sending new messages, and cataloging, filing, and printing messages. These functions are discussed below.

How to obtain Laurel

We recommend strongly that new users obtain a Non-Programmer's disk to be used primarily for processing and managing messages (see section 5 for the reasons behind this recommendation). The BASIC NON-PROGRAMMER'S DISK already has Laurel installed, so if you have obtained a copy of this disk (by the procedure described in the Alto Non-programmer's Guide), you may skip the next paragraph.

Two Alto command files are available on most file servers for obtaining Laurel. If you are a new user, you should issue the following commands to the Alto Executive:

>Ftp FileServer Retrieve <Laurel>LaurelNewUser.cmCR
>@LaurelNewUserCR

This obtains Laurel and starts it up in a tutorial mode. If you are already familiar with Laurel, you should retrieve and execute <Laurel>Laurel.cm, which merely retrieves the files necessary to run Laurel.

Note to Mesa programmers: Laurel 2.0 is a Mesa 4.1 program and therefore requires the Mesa 4.1 version of RunMesa.run to be on your disk. For this reason it can't coexist on a disk with a version of Mesa other than 4.1.

Before using Laurel's hardcopy command for the first time, you must edit your *Laurel profile* (file Laurel.Profile) to declare the name of the printing server to which your hardcopy is to be sent. See section 3.6 for information on the Laurel profile.

How to run Laurel

To invoke Laurel, type

>LaurelCR

to the Alto Executive. This is the default method of invoking Laurel. Command line options are discussed in section 3.5.

To invoke the Laurel tutorial (after obtaining it with @LaurelNewUser), type

>Laurel HelpCR

2.1. The user interface

Laurel is a highly interactive system in which economy and clarity of expression are essential. In striving to provide a convenient user interface, Laurel borrows a number of conventions familiar to users of other Alto subsystems. This section describes the principles that underlie the Laurel user interface.

Command invocation

Most Laurel facilities are invoked by use of the mouse, with the keyboard being used almost exclusively for text entry. Commands are generally represented by words or phrases on the display screen. You invoke a displayed command by moving the mouse until the screen cursor points at the desired command, then clicking a mouse button (usually RED). If you hold down RED and move the cursor to point at a command, the command will appear *inverted* (i.e., white letters on black background). When you release the button, the command name will appear *grayed* (i.e., black letters on a gray halftone background). If, while holding down the mouse button, you change your mind about the command you intend to select, simply move the cursor until the inverted name is restored to its normal state, then release the button. A fine point: position the cursor so that it *points at* the command, not so that it rests on top of it.

Normally, you use RED to invoke commands. Some commands have such a significant effect on the current state that you must confirm them explicitly, and in such cases Laurel will prompt you. If you are certain in advance, you may invoke the command by clicking BLUE. Laurel will then suppress the prompt for confirmation and execute the selected command immediately.

Laurel prompts you for confirmation by displaying the message

Type ESC to confirm, DEL to cancel command.

in the feedback region at the bottom of the screen. It also displays a large flashing question mark in the cursor. You may confirm by typing ESC, CR, Y, or space, or by clicking YELLOW. If you type DEL or N, the command will be aborted. Ultra-fine point: if you change your mind after pressing YELLOW, but before releasing it, clicking RED will turn it into a DEL!

Type-in conventions

Laurel observes most of the standard Alto type-in conventions. When it expects you to supply text, it displays a blinking caret at the appropriate place on the screen. Striking BS or A^c deletes the character to the left of the caret; W^c deletes the word to the left of the caret. Type-in may always be terminated by typing ESC; sometimes other characters will terminate input as well (see below). Commands that do not permit you to make a secondary selection may also be terminated by clicking YELLOW.

Brackets

Several Laurel commands require text arguments. In particular, commands that manipulate a file need to know the name of the file on which they are to act. Text arguments appear in {brackets} following the command name on the screen.

When you invoke such a command with RED, Laurel prompts you to fill in the brackets by displaying a blinking caret within the brackets. If there is already text within the brackets, the caret follows the text. In addition to the type-in conventions described above, the following rules apply: If you type ESC, CR, TAB, or space, or click YELLOW, the caret disappears and the contents of the brackets remain unchanged. If you type BS or W^c the contents of the brackets will be edited as if you had just typed that text. If you type anything else (except as described above), it replaces the complete text within the brackets. Type-in is terminated by ESC, CR, TAB, space, or YELLOW. Typing DEL aborts the entire command.

If you invoke the command with BLUE rather than RED, Laurel executes the command immediately, using the text argument already contained within the brackets, with no further action on your part. This is true of all commands that have text arguments in brackets, with the exception of the user command.

2.2. The display

Laurel maintains four regions on the display screen. From top to bottom they are: the *table-of-contents region*, the *message display region*, the *composition region*, and the *feedback region*. The table-of-contents region holds a directory of the messages residing in the current mail file. The message display and composition regions are used to examine messages received and to compose messages to be sent, respectively. The feedback region displays various information (frequently error notifications) appropriate to particular Laurel commands. Each region will be discussed in detail in subsequent sections.

The display also has three *command menus*. The topmost menu, just above the table-of-contents region, contains a number of miscellaneous commands and status information. The commands, like all commands on the Laurel screen, appear in bold-face type; the status information is in normal

type. We call this simply the *upper menu*. Below the table-of-contents region and above the message display region is the *selection menu*. The commands in this menu are used to manipulate selected entries in the table-of-contents. Below the message display region and above the composition region is the *composition and delivery menu*. As the name implies, this menu contains commands that facilitate the composition of new messages and responses to old ones. The composition region is separated from the feedback region by a single horizontal line.

Scrolling and thumbing

The table-of-contents region, the message display region, and the composition region each have a *scroll bar* in the margin to their extreme left. Within this scroll bar, the RED and BLUE mouse button behave as they do in Bravo. The cursor will appear as a double-headed arrow when positioned in the scroll bar and will change to point up or down when RED or BLUE is pressed.

Thumbing differs substantially from the Bravo style. At the top of each of the scrollable regions is a horizontal bar that separates the region from the menu above. When the cursor is positioned just below this bar and YELLOW is depressed and held, a short vertical line segment appears on the bar. Moving the mouse to the left or right causes a corresponding movement in this line segment. The position of the line segment relative to the left edge of the horizontal bar identifies a location within the text associated with the region. Thus, moving the line segment to the extreme right edge of the bar identifies the end of the associated text; positioning it in the center of the bar specifies the middle of the text. When YELLOW is released, the identified position is brought to the top of the region.

A portion of the thumbing bar appears as a dashed line, whose position and length correspond to the text displayed in the window. Thus, positioning the line segment at the left edge of the dashed line identifies the beginning of the currently displayed text. (Releasing YELLOW at this point naturally has no effect on the display.)

As you will see shortly, it is possible to *select* entities within the table-of-contents and composition regions. The position of the selected entities in each region is indicated on its associated thumbing bar by a short vertical line segment. This segment is always present, and should not be confused with the line segment that appears when you depress YELLOW. You can therefore obtain the effect of Bravo's Normalize command by pressing down on YELLOW and positioning the cursor so that it coincides with the permanent segment on the thumbing bar. Releasing the mouse button then causes the selected text to be moved to the top of the region.

Adjusting region sizes

You can adjust the boundaries of the three major regions using the small squares at the upper right-hand corner of the two lower menus. Point the cursor at the desired box, then press down and hold YELLOW. By moving the mouse up or down, you drag the box with you to a new position on the screen. When you release the mouse button, the menu will move to the new position, and the contents of the adjacent regions will be adjusted accordingly. A fine point: you may find it more convenient to hold down YELLOW and move the cursor to the vicinity of the box. The box will capture the cursor when it comes sufficiently close.

2.3. The table-of-contents region

This region provides an index to the messages in the current mail file. Each entry in the index is numbered and contains the date sent, the sender, and the subject. It is also possible for each entry to have a *mark character* for classification purposes—we will discuss this in section 3.1. Laurel does not permit you to modify the information in the table-of-contents window (except the mark character).

When Laurel begins execution, it normally gets the default mail file and displays its table-of-contents in the top region on the screen. It also places a *selection pointer* (which appears as a small black triangle pointing to the right) next to the last entry in the index (or first unexamined entry, if there is one). This pointer is used to identify messages that you wish to manipulate, and can be repositioned with the mouse buttons. We will discuss it in detail shortly.

2.4. The upper menu

The upper menu contains some obvious status information: the date and time, the version of Laurel you are running, and the amount of free space remaining on your Alto disk. There is also a space reserved for posting the status of your in-box—see section 3.4. Of the remaining items, which are all commands, we discuss only Mail file and New mail here. The other commands are documented in sections 2.8 and 3.3.

Mail files

Laurel maintains one or more *mail files* for you. You should think of mail files as folders in which you organize the messages you receive. When Laurel is first started, it normally checks to see if you have a file folder labelled "Active.mail", and if not, it creates an empty file and labels it for you. This file is called your *default mail file* and is generally used to contain new messages that you have not yet processed. You will see how additional mail files are created in section 3.1.

To direct Laurel's attention to a mail file other than the current one, proceed as follows. Point the cursor at Mail file in the upper menu and click RED. A blinking caret will appear in the brackets following the command, inviting you to type the name of the file you wish to examine. (See the description of {brackets} in section 2.1.) After you have supplied the file name, Laurel will fill the table-of-contents region with the entries from the designated file. You can then use the facilities described in subsequent sections to manipulate this file.

Mail file {file} always reads from the file whose name is displayed in the brackets. However, it observes two conventions that simplify type-in. First, if the displayed name does not contain a period, Laurel implicitly appends ".mail" to the displayed name before accessing the file. Second, if Mail file is invoked with BLUE rather than RED, Laurel omits the prompt for a file name and uses the name currently displayed.

A fine point: Laurel acts on deletions (see section 2.5) whenever either Mail file or Quit is selected. Thus, you cannot Undelete messages in a mail file once you have switched Laurel's attention to a different file.

New mail

Messages that are waiting for you reside in your *in-box*. The *in-box* is also called the *mail box*, but we will avoid this term to prevent possible confusion with *mail file*. You instruct Laurel to move the contents of your in-box to your current mail file by pointing the cursor at the New mail command in the upper menu and clicking RED. The command will then appear on a gray background and the cursor will assume the shape of an hour-glass, confirming that Laurel is busy shuffling messages. You will also observe the free page counter changing. When all messages have been transferred, the gray background and hour-glass will disappear, and the table-of-contents window will be updated to reflect the new messages placed in your mail file. Transferring the contents of your in-box to your mail file renders the in-box empty.

As the New mail command is completed, the table-of-contents region is scrolled so that as much as possible of the new mail is displayed, and the selection pointer is set to point at the first new message. You will also observe that each new entry has a "?" to its left—this indicates that the

contents of the associated message have not yet been examined. The "?" is a *mark character*; you can change it to provide a (primitive) classification of the message—see section 3.1.

2.5. The selection commands menu

Displaying messages

Normally, to examine the contents of a message, you must first select its table-of-contents entry with the selection pointer (see below). However, after obtaining new mail from your in-box, Laurel automatically selects the first new message for you. To display a selected message, simply point the cursor at the **Display** command in the selection commands menu and click **RED**. The selected message will then appear in the display region just below the menu. You may scroll the message or adjust the boundaries of the region for more convenient reading.

To examine the next message listed in the table-of-contents, click **Display** again. The selection pointer will be moved to the next entry, and the text of the message will be displayed. Thus, although Laurel permits you to move the selection pointer explicitly (see below), you need not do so. Simply click **Display** repeatedly and Laurel will advance the selection pointer by one entry each time.

A fine point: Laurel will skip over deleted messages (see below) when advancing the selection pointer. To display a deleted message, you must select it explicitly, then click **Display**.

Selecting messages

Entries in the table-of-contents region are selected in the same way that lines of text are selected in Bravo. Position the cursor to the right of the scroll bar next to the entry you wish to select. The cursor will appear as a right-pointing arrow when it is properly positioned to change the selection pointer. Click **RED**. The selection pointer will move to the indicated entry. By performing explicit selections, you may examine the contents of your mail file in any order you wish.

More generally, you can select any consecutive group of entries in the table-of-contents. This is not particularly useful if you only wish to display messages, but can be convenient if you wish to delete a group of messages (see below), move a group of messages to another file (section 3.1), or generate hardcopy (below). You first select a single message, as described above, then extend the selected range by moving the cursor either up or down and clicking **BLUE**. Thus, selections are made with **RED** and **BLUE** in much the same way they are in Bravo. **YELLOW** has no effect.

Deleting and undeleting messages

After examining some of the messages in your mail file, you may wish to delete them. The **Delete** command, when activated by pointing the cursor at it and clicking **RED**, will cause a line to be drawn through all *selected* entries in the table-of-contents. You may find it convenient to use **Display** and **Delete** alternately when processing newly-arrived junk mail.

If you discover that you have inadvertently deleted some messages that you want to keep, simply reselect them if necessary, position the cursor over **Undelete**, and click **RED**. The lines drawn through the table-of-contents entries of the selected messages will be removed.

Deleted messages are expunged (i.e., removed permanently) from the current mail file whenever a **Quit** (see section 2.8) or **Mail file** command is executed.

Generating hardcopy

To print a copy of one or more messages in your mail file, select the message or messages in the table-of-contents, point at **Hardcopy**, and click RED. Laurel will generate hardcopy of each message in the form of an inter-office memorandum, one page (or more if necessary) per message, and send it to be printed on your default printing server. Messages that have been deleted (i.e., whose table-of-contents entries have lines through them) will not be printed, even if they are selected.

There are no options that you can exercise at the time you issue the **Hardcopy** command. However, by changing your *Laurel profile*, you can control certain aspects of the hardcopy formatting as well as a few other things such as the name of your printing server (see section 3.6).

2.6. Composing and delivering messages

Composition is the process of building the header and body (see the beginning of section 2) of a particular message. *Delivery* is the process of transmitting a composed message to its specified recipients. Laurel separates these two actions and supplies distinct commands appropriate to each in the composition and delivery menu.

Laurel provides four ways to initialize the content of a message. You may compose a *new* message, or *answer* one you have received, or *forward* an existing one to a new recipient, or *get* a previously-composed form or message from a file on your Alto disk. In the composition and delivery menu (below the message display region), there are four commands corresponding to these actions: **New form**, **Answer**, **Forward**, and **Get**. Simply select the one you wish by positioning the cursor appropriately and clicking RFD. If the composition region contains an undelivered message, these commands request confirmation before constructing a new form. As in other such contexts, selecting these commands with BLUE automatically supplies confirmation. Even after confirmation, Undo (see below) can still be used to recover the previous contents of the region.

It is important to understand that **New form**, **Answer**, and **Forward** only provide a message *form*; you must edit the form before requesting that it be delivered. **Get** will be described in more detail later (section 3.2).

Composition

New form gives you a new message form in the composition region. The form contains "To", "Subject", and "cc" fields, which you should fill in as appropriate. Laurel provides a subset of Bravo's editing commands to permit you to compose these fields and the message body. Laurel provides the Append, Delete, Insert, Replace, ESC (repeat) Undo, and secondary selection functions of Bravo, but does not handle italics, bold-face, multiple fonts, or any "looks". For important additional editing functions, see section 3.2. In short, the message you compose is simply an unformatted text string. However, Laurel does supply automatic line breaks as Bravo does, so you need not type CRs except to produce white space.

At present, the Laurel editor is not quite compatible with Bravo, though it may be in the future. In particular, you should be aware of the following differences. An Undo will undo the previous command. Undo always undoes the previous command *exactly where it was given*. To move text from one place to another use Delete followed by Insert ESC. **New form**, **Answer**, **Forward**, and **Get** may also be undone.

In general, Laurel imposes format restrictions only on the message *header*. By definition, the header ends at the first blank line (i.e., two successive CRs). You should therefore be careful not to delete the blank line provided by Laurel in each of the three initial forms. The header itself consists of a sequence of fields, some of which are required to be present. When composing a message you should always fill in the "To" field and the "Subject" field; you may delete the "cc" field if it is not needed. When the message is ultimately delivered (see below), Laurel will supply your name and

the date, so you need not include them explicitly.

Answer and **Forward** initialize a message form in different ways, but both take information from the message *currently displayed in the message display region* (which is not necessarily the one selected in the table-of-contents region). **Answer** fills in the "To" field with the sender of the currently displayed message and sets the subject to be "Re: *sender's subject*". It also sets the "cc" field to include all of the recipients of the message being answered. If you don't like these substitutions, you may, of course, change them using the editing facilities. **Forw.rd** copies the message body from the display region into the composition region. After clicking either **Answer** or **Forward**, you must complete the message by editing any remaining uninitialized fields in the message header and body.

Laurel identifies fields that it expects you to replace by supplying a keyword bracketed by black rectangles, e.g., ■Recipients■. Laurel will refuse to deliver any message whose header contains one of these fields. This protects you from simple oversights, such as forgetting to supply a "Subject" field.

Delivery

Once you have composed the message you wish to transmit, you may initiate its delivery to the recipients by pointing the cursor at **Deliver** and clicking RED. Laurel will fill in your name and the date (though they won't appear in the composition region) and proceed to send the message. A gray background will appear behind the **Deliver** command and the cursor will change to an hour-glass. If Laurel discovers an error in the list of recipients, it will give you an opportunity to cancel the delivery and correct the mistake. When the list is acceptable (i.e., all specified recipients are known to have in-boxes), Laurel will deliver the message and remove the hour-glass and gray background. After successful delivery, the word **Deliver** will disappear and will be replaced by "*delivered*". If an error occurs during delivery, an explanation of the error condition will be displayed in the feedback region. You may cancel delivery while the message "*Type DEL if you wish to cancel delivery*" appears in the feedback region.

Before filling in your name in a composed message, Laurel will check to see if the message header already contains a "From" field. If so, Laurel inserts a "Sender" field with your name and leaves the "From" field untouched. It is the "From" field that is normally displayed in the table-of-contents. The name Laurel furnishes is your logged-in user name—see section 3.3.

Deliver tells you the number of recipients to whom it will deliver the message. If this number exceeds 30, you must confirm the delivery. This is intended to minimize unintentional deliveries to large distribution lists. **Deliver** also tells you the size of the message (in characters).

2.7 The feedback region

Laurel uses the feedback region for three classes of information: status reports, exceptions, and confirmation requests. Status reports are displayed by various commands, e.g., **Deliver**, to report circumstances of interest to you but which require no direct action on your part. Exceptions are notifications of errors committed by you (or Laurel) and are flashed to alert you that some corrective action is probably required. Confirmation requests (see section 2.1) flash both the feedback region and a "?" in the cursor, alerting you to the need for immediate action before Laurel can continue.

2.8 Leaving Laurel

To exit from Laurel and return to the Alto Executive, point the cursor at **Quit** (in the upper menu) and click **RED**. Laurel will prompt you for confirmation. After you confirm, Laurel will act on the deletions indicated in the table-of-contents region, eliminating all messages from the mail file that have lines drawn through their table-of-contents entries. If you invoke **Quit** with **BLUE**, Laurel will omit the confirmation prompt. When you re-enter Laurel at a later point, these messages will no longer appear in the table-of-contents. They are gone forever.

Leaving Laurel by any means other than **Quit** is not recommended and will slow down subsequent re-entry to Laurel.

3. Additional facilities

3.1. Filing and classifying messages

Laurel provides two facilities to assist you in classifying your mail. Remember that your default mail file should be viewed as holding those messages upon which you have yet to act. Naturally, you may want to file away those messages on which you have *already* acted. The command **Move to {file}** in the selection menu allows you to do this. *Mark characters* provide a simple means of attaching labels to individual messages within a mail file.

Move to {file}

To invoke **Move to {file}** proceed as follows. First, select (with selection pointers—see section 2.5) the messages you wish to transfer to a separate file. Next, point the cursor at **Move to** and proceed as you would for **Mail file** (see sections 2.1 and 2.4). After you have supplied the file name, Laurel will append the selected messages to the indicated file. Laurel will draw a line through the table-of-contents entries of the messages that have been moved so that the corresponding messages will be deleted when you leave Laurel (or change mail files). If you wish to retain any of these messages in the current mail file *as well*, you may use "Undelete" as described above. Laurel will also *mark* (in the mark character field—see below) moved messages with an "m".

Move to does not move any messages that are indicated as deleted. If a range of messages containing both deleted and non-deleted messages is selected for **Move to**, only the non-deleted messages are moved.

Move to observes the same type-in conventions that **Mail file** does. It also implements one other convention that guards against misspelled file names: if the displayed file name does not correspond to any existing file, Laurel prompts you for additional confirmation before creating and writing a new file with that name.

Mark characters

You may wish to classify messages within a particular file. Laurel provides a primitive marking system by allowing you to supply a single *mark character* for any table-of-contents entry. To set the mark character, position the cursor to the left of the desired entry but to the right of the line bar and selection pointers. Click **RED**. A blinking caret will appear, inviting you to type a single character. This character will be retained in the table-of-contents entry and displayed whenever the entry is in the region. It will also be retained in the message if it is moved to another file (using **Move to {file}**). Note that "?" and "m" are valid mark characters but have additional semantics assigned to them by Laurel. Other mark characters may be assigned special meanings in the future, e.g., to permit selective display of the table-of-contents. You may change the mark character by positioning the cursor over the mark to be changed and repeating the preceding steps. A space is a valid mark character.

3.2. Additional editing and delivery facilities

Secondary selection

Laurel supports secondary selection (for Insert, Append, and Replace commands) in the same manner as Bravo does. That is, you may identify the text string to be inserted by using the mouse buttons to select characters, words, lines, or paragraphs. You may select text in either the message

display region or the composition region, and you may scroll or thumb within either region while making the secondary selection.

Paragraphs

Although Laurel does not implement Bravo's notion of paragraphs, it does provide a selection mode that gives similar results in simple cases. Laurel defines a paragraph terminator to be two successive CRs. You may select a paragraph by positioning the cursor in the line bar to the left of the desired text and clicking YELLOW, just as in Bravo. The main visible difference is that Laurel will not display Bravo's paragraph symbol in the cursor.

Distribution lists

Suppose you have a collection of people to whom you frequently send messages. To avoid having to type the entire list of names every time you send a message to the group, you may proceed as follows. Create an Alto file containing the list of names (separated by commas, no CR at the end). You may find it convenient to "label" such distribution lists; you may do so by prefixing the list with identifying text followed by a colon, and then terminating the entire list with a "matching" semicolon. For example, such a file might contain

Alpha users: Brown, Doe, Jones, Smith;
Name this file "AlphaUsers.dl".

Now, when composing a message in Laurel, insert the file name in the appropriate field of the message header (e.g., following "To" or "cc"), followed by an up-arrow character:

To: AlphaUsers[↑]

While delivering the message, Laurel will read the file to determine the recipients. Multiple distribution list specifiers may be used within a single field and may be intermixed with names of individual recipients—separate them all with commas. If you give your distribution list files the extension ".dl", you may omit typing the ".dl" to Laurel.

You need not worry about a recipient appearing in more than one distribution list. Laurel will detect duplicate names among the recipients and ensure that each receives the message only once. Thus, you may use multiple distribution list specifiers freely within the "To" and "cc" fields of the message header.

The present distribution list mechanism is a temporary one, and has some inconvenient aspects. In particular, since distribution lists are presently stored on your own Alto's disk, they can get out-of-date rather easily. Commonly-used distribution lists are maintained centrally, presently as files <Secretary>*.dl on Maxc. We have tried to streamline the task of keeping distribution lists current by providing a command file, DLUpdate.cm, which updates local distribution lists from the central source. Registered Laurel users (i.e., those listed in LaurelUsers.dl) are notified when DLUpdate.cm should be run. DLUpdate.cm is automatically retrieved by Laurel.cm and LaurelNewUser.cm (see section 2). Note that DLUpdate.cm updates *only* those distribution lists that you *already* have on your disk. To obtain a copy of a distribution list you don't already have you must use FTP to retrieve it from the <Secretary> directory on Maxc. Thereafter, running DLUpdate.cm will keep it up-to-date along with all the others.

Get and Put

The composition and delivery menu includes the commands Get, Put {file} for filing and retrieving the contents of the composition window. Put writes out the entire contents of the composition window onto the file whose name is inside the brackets. (If a file with that name already exists, Laurel requires an extra confirmation before overwriting it.) Get replaces the contents of the composition window with the text contained in the file. As usual, if you click RED over Get or Put, a blinking caret appears within the brackets and you may fill in or edit the file name before confirming the command, whereas if you click BLUE, Laurel will execute the command immediately using whatever name is already inside the brackets.

Put is useful when you want to save a partially-composed message to be completed and delivered later. You may also want to do this if you encounter persistent network or mail server problems upon attempting to Deliver. **Get** is also handy for obtaining custom forms (or form letters) that you have prepared previously. Obviously, you can use Laurel to edit arbitrary text documents in much the same fashion as Bravo; however, since Laurel does not provide any of Bravo's formatting capabilities (multiple fonts, looks, etc.), you will find Laurel to be of limited utility in this respect.

In addition to the **Get** and **Put** menu commands just described, there exist Get and Put keyboard commands. These work somewhat differently from the menu commands. In particular, whereas the **Get** and **Put** menu commands operate on the entire composition window, the Get and Put keyboard commands operate only on the *current selection*. **Get** and **Put** do, however, make use of the file name contained in the **Get**, **Put** {*file*} menu command.

The **Put** keyboard command first prompts you either to confirm the file name already in the brackets or to type in a new file name, then writes a copy of the *currently selected text* onto the file by that name. This text is not removed from the composition window.

When you issue the **Get** keyboard command, Laurel first puts a copy of the *currently selected text* into the file name brackets. You may now either confirm that file name (with **ESC** or whatever) or type some other name followed by **ESC**. Laurel then *replaces* the currently selected text by the contents of the file, leaving the replacement text selected.

The **Get** keyboard command can be used as a file substitution device anywhere within a composed message. To include the contents of a file within a message, insert the file name (including the extension) at the desired point. Then select the file name with the mouse buttons as you would for a Replace command, but type Get ESC instead of Replace. Laurel will replace the file name with the contents of that file.

It is sometimes useful to expand a distribution list *before* sending a message. You can do this by inserting the distribution list name in the appropriate header field, selecting it, then typing **Get**. In addition, there is a short-cut that permits you to omit the explicit ".dl" extension. If you type the file name followed by a "+", select it, and type **Get**, Laurel will append the ".dl" (if no extension is present) and perform the **Get**. You may take advantage of this feature to peek at the contents of a distribution list: use **Get** as just described, then undo the expansion by typing **Undo**.

Put produces a file that can be read and manipulated by Bravo. In fact, you can create a Bravo document, use the **Get** keyboard command to read it into a composed message, then send the message. The recipient can perform a Forward to move the received message into the composition window, select the portion of the message consisting of the Bravo document, and issue the **Put** keyboard command to write it onto a file. It should be noted that Bravo formatting information in the document may contain unprintable characters, which are displayed by Laurel as black rectangles. However, Laurel does not actually alter these characters, so formatting information is not lost. You must take care to Put the entire Bravo document (and nothing else), or else Bravo won't be able to read it.

Message header format

Laurel supports a subset of the ARPA standard for message headers (RFC 733/NIC 41952). Happily, users need not be aware of most of the requirements of this extensive standard. However, two features should be noted. First, comments may be included in a header item by surrounding them with parentheses. Second, recipients at Arpanet sites other than Parc-Maxc may be specified as either "*name* @ *site*" or "*name* at *site*".

You should avoid typing extra CRs in message headers, since the ARPA standard specifies some less-than-obvious behavior in such cases. In practice, this is rarely a problem, because Laurel will supply line breaks for you (both on the screen and in the transmitted message).

3.3. Authentication and logging In

In order to perform New mail, you must be an *authenticated user*. This means your name and password must be accepted by a centralized authenticator (currently Maxc). When you start up Laurel, it obtains your name and password from the Alto Operating System and submits them to the authenticator for verification. If the authentication fails, the message "Name or password invalid" will appear in the upper menu, just below the Laurel version number. Laurel will not allow you to receive messages until you have been authenticated successfully.

To enter a new name and password, point the cursor at User in the upper menu and click RED. Laurel will invite you to supply a user name in the brackets following User. Respond as you would for other {brackets} prompts (see section 2.1). After you have entered your user name, Laurel will prompt you for a password in a similar way. You must supply the password *explicitly*—ESC, YELLOW, etc. cause a null password to be entered. Laurel will not display your password as you type it. A fine point: At present, the name you supply to the User command is used as the directory on Maxc where your mailbox is located. This may not be true in the future, but it will always be possible for Laurel to locate your mailbox from the user name you supply.

Laurel normally requires that you be an authenticated user before it will allow you to receive or send messages. If, for some reason, you are not an authenticated user and you select Deliver, Laurel will prompt you for confirmation. If you confirm, Laurel will append the phrase "(not authenticated)" to the sender identification it includes in the outgoing message. This makes it difficult for others to masquerade as you (or vice-versa).

When you leave Laurel, your name and password are passed back to the Alto Operating System.

3.4. Polling for new mail

Laurel checks your in-box periodically to determine if new mail has arrived. If it finds messages waiting in your in-box, it stops polling and displays "You have new mail" below the Laurel version number at the left-hand side of the upper menu. You may then use the New mail command (see section 2.4) to move the messages in your in-box to the current mail file. After you have done so, Laurel will remove the "You have new mail" message and resume its periodic polling.

Because this polling is completely automatic, you can leave your Alto running Laurel when you are not using it for other things, and glance at the screen occasionally to see if new messages have arrived. The polling interval at present is about 5 minutes.

3.5. Command line options

Laurel has several command line options that can be specified when you invoke it from the Alto Executive.

Mail file selection

Laurel's default action, triggered when you type

>LaurelCR

is to perform an implicit Mail file on Active.mail. If you type

>Laurel filenameCR

Laurel will read *filename.mail* instead. If you supply an explicit extension, Laurel will respect it, otherwise it will use ".mail".

In-box interrogation

You can also request that Laurel access your in-box automatically when it begins execution. Type
>Laurel/nCR or >Laurel/n filenameCR

Laurel will first perform an implicit Mail file on the indicated or defaulted file (see above). It then will check to see if messages are present in your in-box. If so, Laurel performs an implicit New mail. Upon the completion of the New mail (if any), Laurel is available for normal use.

You may also invoke Laurel by

>Laurel/cCR or >Laurel/c filenameCR

If messages are present in your in-box, Laurel will behave as though "/n" were specified. If no new mail exists, Laurel will omit the implicit Mail file and return directly to the Alto Executive. A fine point: if Laurel is unable to determine whether your in-box has messages, it ignores the /c switch.

Send message mode

If you merely want to send a message, you may request that Laurel omit its implicit Mail file when it begins execution. Type

>Laurel/sCR

Instead of reading a mail file, Laurel will perform an implicit New form as it starts up. You may then compose and deliver your message in the usual way. If, after doing so, you wish to process a mail file, simply invoke Mail file and supply a file name as described in section 2.4.

3.6. The Laurel profile

Laurel obtains certain configuration information and options from your *Laurel profile*, contained in file Laurel.Profile on your Alto disk. A standard Laurel profile is installed when you obtain Laurel using the LaurelNewUser.cm or Laurel.cm command file (section 2). The only things in the Laurel profile you are likely to want to change at present are the name of your printing server and perhaps some of the hardcopy format options. For completeness, however, nearly all possible options are documented here.

Laurel.Profile is a text file containing a sequence of *parameter lines*, each of which has the following form:

name: value^{CR}

All lines in the file must adhere to this format. This implies that there may not be any blank lines in the file. Furthermore, the last character in the file must be the CR of the last parameter line. Bravo formatting is *not* permitted. You should probably use only Laurel to edit Laurel.Profile. Note, however, that Laurel configures itself according to Laurel.Profile when it is started up, so after editing Laurel.Profile you must Quit and restart Laurel before the changes will take effect. The Laurel program processes this file very quickly, with only minimal error checking. If Laurel.Profile is not properly formatted, the only indication from Laurel will be the exception message "The file Laurel.profile is missing or bad."

The *value* may be preceded by any number of blanks and begins at the first non-blank character. Subsequent characters may be blank and are included in the *value*. The *value* is terminated by the CR at the end of the parameter line.

A typical Laurel.Profile for use at Parc might be as follows:

Send: Maxc
Retrieve: Maxc
Authenticate: Maxc
Hardcopy: Clover
PrintedBy: \$

Parameter names may be spelled in upper or lower case or in any combination. This example leaves out many possible parameter lines. Parameters that are missing from Laurel.Profile are given default values by Laurel.

Send: *host name*

The name or network address of the mailbox server machine to which you will send all outgoing mail. (Default: Maxc) Network address constants of the form *number#number#* must be used at sites where there is no gateway providing name lookup services.

Retrieve: *host name*

The name or network address of the mailbox server machine from which you will retrieve new mail. (Default: Maxc)

Authenticate: *host name*

The name or network address of your authentication server, i.e., the machine that verifies user name and password combinations. (Default: Maxc) Normally the values specified for Send, Retrieve, and Authenticate will all be the same.

Hardcopy: *host name*

The name or network address of the hardcopy server machine to which all hardcopies will be sent. (Default: an illegal server name)

PrintedBy: *text*

A line of text that will be printed on hardcopy break pages to identify the person making the hardcopy. Any "\$" characters will be expanded to your logged-in name at the time you generate the hardcopy. (Default: \$)

Herald: *text*

A line of text to be printed at the top of the first page of each hardcopied message. (Default: Inter-Office Memorandum). You should take care to ensure that the herald will fit on one line when hardcopied.

HeraldFont: *font*

The font that will be used to print the herald on hardcopied messages. A font value must take the form:

family-name point-size face

Examples:

TimesRoman 14 B

Helvetica 10

The optional B or I following the point size indicates bold or italic face. If the CR immediately follows the point size, the regular face is used. (Default: TimesRoman 12 B)

Logo: *text*

A line of text to be printed at the logo position between the header and the message body on the first page of each hardcoded message. (Default: XEROX). You should take care to ensure that the logo will fit on one line when hardcoded.

LogoFont: *font*

The font that will be used to print the logo on hardcoded messages. The format for the font value is given above under HeraldFont. (Default: Logo 24)

4. If things go wrong . . .

In Laurel, as in most interactive systems, lots of things can go wrong. Also as in most systems, some of them are your fault and some of them are not. Laurel tries to prevent you from wreaking destruction upon your environment, and reports any (perceived) violations in the feedback region.

The error reports in the feedback region are intended to be self-explanatory. If you cannot figure out what one means or what to do next, please send a message to LaurelSupport (using Laurel if possible).

If Laurel detects certain internal error conditions, it interrupts whatever command was in progress and posts a message in the feedback region. In some cases, Laurel may decide that the error should be reported to LaurelSupport, and, in such cases, it will prepare an error report form (after confirmation from you). This form contains internal status information of interest to the Laurel implementers and should be used whenever possible. If you confirm the use of this form, Laurel will restart itself and display the form in the composition region. You should then follow the instructions contained within the form, after which Laurel will again be available for normal use.

5. Laurel and MSG

This section addresses the relationship between MSG and Laurel. As mentioned in section 1, MSG users *must* thoroughly understand the contents of this section *before trying to use Laurel for the first time*. If you don't use Maxc or you have never used MSG, you may skip this section entirely.

Before proceeding to the gory details, we should remind you that these comments apply *only* to Laurel versions 1 and 2. Most of the difficulties described below are artifacts of a temporary implementation of a number of Laurel facilities. Future releases will cure these ills, so if Laurel appears insufficient for your present needs, don't give up.

Mail files

Laurel keeps nearly all of its files on your Alto disk. The only exception is your in-box, which is currently file Message.txt in your directory on Maxc. Getting new mail (with Laurel) moves messages from Maxc to your Alto disk, *emptying Message.txt*. This makes it very inconvenient to continue to use both Laurel and MSG, and it makes Laurel inconvenient to use if you switch Alto disks frequently. For now, our best advice is to designate a disk on which you will use Laurel exclusively, and stick to it. You will need about 500 disk pages to hold Laurel (including RunMesa.run), a moderate-sized mail file, and several distribution lists.

Once you have mail files on your Alto disk, it becomes impossible to process them using MSG, since they are on the Alto. Furthermore, MSG and Laurel maintain mail files in formats that are incompatible. You can continue to use MSG to maintain mail files on Maxc, and you can use Laurel to maintain mail files on your Alto, *but you cannot move mail files back and forth*. If you

are willing to convert completely to Laurel, you can move MSG-constructed mail files to your Alto for use by Laurel, as explained below. *However, this is a one-way street; once on your Alto the files cannot conveniently be moved back to Maxc and processed by MSG.* If you are not willing to convert completely, we recommend that you wait to use Laurel until it is able to deal with remotely-stored mail files.

On the positive side, your correspondents never need to know that you have converted to Laurel, since Laurel is able to process your Message.txt file directly. All that has happened is that you have started using that file in a different way (as your in-box), but the change is invisible to people who send you messages.

Mail file philosophy

Laurel encourages you to use your default mail file (Active.mail) as a *temporary* storage area, not an archive. Accordingly, deleted messages are expunged whenever you leave Laurel or shift your attention to another mail file. There is no analogue of MSG's Quit command (which preserves deleted messages); Laurel's Quit is like MSG's Exit. If you wish to classify messages, use mark characters (section 3.1). Eventually, selective display on the basis of mark characters may be possible. Use separate mail files to obtain an archive facility. Laurel's performance will be better and your screen will be less cluttered.

Overwrite

Laurel doesn't have a command corresponding to MSG's Overwrite. However, the same effect can be obtained by selecting Mail file with BLUE. After you have read sections 2.4 and 2.5, it should be evident to you why this works.

Processing your mail away from an Alto

You can use MSG from any reasonable terminal dialed to Maxc or connected to the Arpanet, but you can't use Laurel without an Alto. Consequently, if you have no Alto available and would like to process your mail, you must use MSG. *If you observe a few conventions while using MSG, you will not need to reprocess your old messages with Laurel.* Thus, you can use MSG from a home terminal or when you are out-of-town, and revert to Laurel when your Alto is again at hand.

As long as you manipulate *only* your Message.txt file with MSG, Laurel will note what you have done when you perform a New mail command. (Remember, Laurel uses Message.txt as your in-box.) Specifically, deleted messages in Message.txt will appear with lines through them (i.e., marked as deleted), and the examined/unexamined status of each message will be reflected in the mark field in Laurel's table-of-contents entry.

Moving mail files from Maxc

Because MSG and Laurel have incompatible mail file formats, you cannot simply move your archival mail files to your Alto disk and expect Laurel to read them. The following (admittedly tedious) procedure should be followed for each mail file:

1. On Maxc, append the desired mail file, say Archive.txt, to Message.txt:
@Append Archive.txt Message.txtCR
2. On the Alto, in Laurel, click RED over Mail file and supply the file name "Archive".

3. Select New mail. After the messages have arrived, ensure that no unexpected ones have been included at the beginning and/or end of the file. (New mail might have arrived on Maxc before or after step 1.) Move any extraneous messages to "Active.mail".
4. Quit Laurel and go back to Maxc for the next file. You may delete Archive.txt on Maxc at this point.

Fortunately, once you have moved all of your files in this way, you won't have to do it again, since Laurel will maintain them in its own format on your Alto disk.

6. Look before you leap . . .

As you become familiar with Laurel, you will doubtless discover features that please you and "features" that annoy you. This section reports some of the annoying "features" other users have encountered and, in some cases, what to do to get around them.

Formatting messages for non-Laurel users

Laurel breaks lines in a transmitted message by a rather simple-minded algorithm. This may lead to line overflow when the message is read on certain terminals using MSG (or other mail systems). If you know that some of your recipients have this problem, you may wish to exercise care in the formatting of your outgoing messages. Use explicit CRs and keep your lines down to about 80% of the Laurel screen width. (A relatively painless way to do this is to compose the entire message *without* using CRs, then just before delivery make a final pass over the message, substituting CRs for spaces at appropriate points.)

Secondary selection from the message display region

Text in the message display region generally has real CRs in it. Laurel respects this formatting information, even when displayed text is copied into a composed message. If you don't like what you get when you insert text from a displayed message, you will have to replace explicit CRs by spaces before Laurel will reformat the text.

Tabs

Laurel does not presently implement true tabs. Instead, a tab is converted to a fixed amount of white space. Thus, if you use tabs only at the beginning of a line, things will line up. In other cases, you are out of luck.

The Answer form

The form provided by the Answer command is not always exactly what you might want. Laurel takes the viewpoint that it is easier to delete than to add text, and therefore includes all the information that seems to be relevant. For example, Laurel often includes your name in the "cc" field of the answer form, even though you may not want a copy of the message you are composing. You should *not* expect that the answer form will be exactly right; always examine the header to be certain it contains the desired information.

In particular, if you reply to a message that was directed to a distribution list, the Answer command will copy the distribution list specifier into the "cc" field of the answer form. You should consider carefully whether or not it is appropriate for your reply to be sent to that distribution list, and if not

delete the distribution list specifier.

File version numbers

Laurel does not support the use of Alto file version numbers. This restriction stems from the underlying implementation of the Mesa run-time system, and is not likely to change. Therefore, mail files and distribution lists should not have version numbers.

Space required for mail files

Some users, when converting from MSG to Laurel, have had a rude shock when they transferred their archival mail files from Maxc to their Alto disk. Recall that one Maxc disk page becomes four Alto disk pages when a file is moved. This is another reason for dedicating an Alto disk to mail processing and filing, particularly if you maintain archival mail files.

Hardcopy usage

Laurel's Hardcopy command is intended for use in printing copies of isolated messages that you may wish, for whatever reason, to obtain in hardcopy form. It is *not* suitable for obtaining hardcopy archives of all your messages, say, to put in your filing cabinet. Because Laurel prints each message on a separate page, using Laurel hardcopy for such archival purposes would produce unwieldy quantities of paper.

If you do wish to maintain a hardcopy archive, we recommend that you use the EmPress subsystem to print your mail file as continuous, unformatted text.

7. Things a casual user doesn't really need to know

Laurel currently executes almost exclusively on your local Alto. Eventually, a large fraction of its facilities will be provided remotely in a transparent fashion, but for now everything except in-box storage is local. This means that all mail files reside on the local Alto disk. For each mail file *x*, Laurel creates a separate file *x-dmSTOC*, which holds various internal information about the table-of-contents. When you invoke Mail file {*x*}, Laurel will recreate this file if necessary. Naturally, this slows down the Mail file command.

Laurel creates scratch files while it is working and leaves them on your Alto disk. You may delete these files (named DMS-*n*.TMP for various values of *n*) if necessary, though Laurel ensures that they will not grow very large. Deleting them lengthens the time required to start up Laurel.

If Laurel generates an automatic error report directed to LaurelSupport (see section 4), it will leave behind a file Laurel.BugReport\$ containing the text of that report. You may delete this file if you wish.

In-boxes are *not* stored locally. Laurel assumes that your in-box resides on the server machine indicated in the "Retrieve" line of your Laurel profile, and uses that machine's mail server to read and empty them. Messages sent via the Deliver command are forwarded to the mail server specified in the "Send" line of your Laurel profile. Arpanet addresses, specified as "*name* @ *site*" or "*name* at *site*", will work correctly only in messages delivered to Maxc. All this is strictly temporary—eventually, Laurel will understand about mailboxes in multiple places and will introduce its own notions about naming message recipients.

8

Printer Dover

Spruce version 11.0 -- spooler version 11.0

File: subsystemsCATALOG.BRAVO

Creation date: September 18, 1979 7:42 AM

For: falis

14 total sheets = 13 pages, 1 copy.

ALTO SUBSYSTEMS CATALOG

June 1979

Filed on [Maxc]<AltoDocs>SubsystemsCatalog.press.

This catalog lists and briefly describes the various Alto subsystems. A subsystem is defined to be any program that runs on the Alto whether under control of the standard executive or as a stand-alone. Each subsystem has an entry in this form:

ProgramName:	Description
DOCUMENTATION:	FilePath

Italicized *ProgramNames* indicate entries that are new or significantly altered. To simplify locating a piece of software to perform a specific function, a functional cross reference of Alto subsystems is provided beginning on page 8. New subsystems are continually being developed throughout the Whole Alto World. This catalog is maintained and distributed by the Whole Alto World Coordinator (Ron Cude). If you have or know of a subsystem which you feel should be included, please provide the above information to the Coordinator. The following criteria should be met before a subsystem is cataloged and distributed:

1. The program should be in general use at the installation.
2. Support should be provided to at least fix major bugs.
3. Satisfactory documentation must be available and up to date.
4. Management must indicate the sensitivity of the item to outside disclosure.

The subsystems listed are generally available from your local File Server or Maxc under the <Alto> directory; the supporting documentation is found in the <AltoDocs> directory unless otherwise noted. If you do not have access to a File Server or Maxc, contact the Coordinator.

AIS: A driver subsystem which interacts with the user to perform a set of standard operations on imaginal data stored as AIS (Array of Intensity Samples) files.
DOCUMENTATION: AIS-Manual.press.

AISdump: A part of the AIS system to write out to the Diablo disk the decimal values for pixels within a specified window.
DOCUMENTATION: None.

AISMagnify: A part of the AIS system to magnify or minify AIS format images in either 1 or 8 bit/pixel form.
DOCUMENTATION: [Erie]<AIS>Memos>AISMagnify.press.

AISshow: A part of the AIS system that displays an AIS image file on the Alto display.
DOCUMENTATION: [Erie]<AIS>Memos>AISshow.press.

ANALYZE: A part of the Design Automation System that transforms logic diagrams produced using SIL into a file which can be input to the GOBBLE wirelister.
DOCUMENTATION: [Maxc]<SIL>SilManual.press.

APROM: Superseded by PROM.

ASM: An assembler for the Alto machine language which produces relocatable files compatible with the BCPL loader, BLDR.
DOCUMENTATION: ASM.tty or Subsystems.press.

BCA: A basic cross-assembler for micro-computers.
DOCUMENTATION: BCA.press.

BCPL: A compiler for Alto BCPL language which produces files relocatable with the BLDR loader.
DOCUMENTATION: BCPL.tty/.press.

BLDR: A loader for the relocatable files produced by BCPL and ASM.
DOCUMENTATION: BCPL.tty/.press.

BRAVO: A text editor having extensive formating and hardcopy facilities.
DOCUMENTATION: ALTO User's Handbook, BRAVO Course Outline, BRAVO.press and BRAVOSUMMARY.press.

BTREESTEST: A B-Tree dictionary maintenance program used to support the PROOFREADER data base.
DOCUMENTATION: ProofReader.tty.

BUILD: a part of the Design Automation System that helps with the data management aspects of building boards and keeping the design automation data files current.
DOCUMENTATION: [Maxc]SIL>SilManual.press.

BUILDBOOT: a program for constructing type B bootfiles from either an executable (BLDR out) file or a segment file.
DOCUMENTATION: BUILDBOOT.tty or Subsystems.press.

CALCULATOR: a bootfile that pictures a TI SR-52 on the display which is operated by using the mouse to select the appropriate keys. It is not programmable.
DOCUMENTATION: SR-52 Manual.

CALLFTP: a subset of FTP which always lives in the operating system. CALLFTP is much smaller than FTP and can be used when space is tight.
DOCUMENTATION: FTP manual.

CHAT: a program for establishing PUP Telnet connections between a pair of cooperating parties. Its chief function is to permit Alto users to talk to Maxc.
DOCUMENTATION: ALTO User's Handbook, CHAT.tty or Subsystems.press.

CLEANDIR: a program to garbage collect a disk file directory (but not disk space).
DOCUMENTATION: Subsystems.press.

CONDENSE: a program to retrieve the screen bitmap from the SWAT and SWATEE files for display or output to an AIS or Press format disk file.
DOCUMENTATION: Menu is self-explanatory.

COPYDISK: a program for copying entire diskpacks. It will copy from one drive to another on the same machine, or between drives on separate machines via a network using Diablo Model 31/44 and Trident T-80/T-300 disks.
DOCUMENTATION: COPYDISK.tty or Subsystems.press.

COPYFROMDRIVE1: a program to copy an individual file from DP1 to DP0 of a dual disk Alto.
DOCUMENTATION: Subsystems.press.

CREATEFILE: a program to create a file of a given size, attempting to allocate it on consecutive disk pages.
DOCUMENTATION: Subsystems.press.

CRTTEST: a diagnostic program used to adjust the Alto display linearity. Three different sized grids are displayed in rotation (press any key to change grids).

DOCUMENTATION: none.

CRUMBLE: a program to compress and, optionally, encrypt data files. The resulting files can be stored or transmitted but must be expanded and decrypted before processing by Alto programs.

DOCUMENTATION: Crumble.press.

DDS: a program to manage an Alto diskpack. Facilities are provided to display filenames, lengths, creation-read-write dates, and contents, internal operations such as delete, and rename, and external operations such as Send and Execute.

DOCUMENTATION: ALTO User's Handbook or DDS.tty or Subsystems.press.

DMT.BOOT: a memory diagnostic and statistics gathering program.

DOCUMENTATION: DMT.tty or Subsystems.press.

DO: a wonderful program giving a parameterized interface to the executive. More powerful than CM files, and useful with the IF program.

DOCUMENTATION: DO.press.

DPRINT: a program to type text files on the Diablo HyType printer.

DOCUMENTATION: DPRINT.tty or Subsystems.press.

DRAW: an interactive illustrator program for creating black-and-white or color pictures composed of lines, curves, and text captions. The illustrations can be output to a one page press file.

DOCUMENTATION: ALTO User's Handbook plus DRAWnews.press, an on-line manual (part of the DRAW package), and DRAW-Summary.press.

EDP: an Ethernet interface diagnostic.

DOCUMENTATION: None.

EFTP: a small but inefficient Ethernet FTP protocol.

DOCUMENTATION: EFTPPackage.tty.

EMPRESS: a program to send press and text, e.g. bravo format, files to a press printing server. Simple formatting options such as Tab and FormFeed are available.

DOCUMENTATION: EmPress.tty or Subsystems.press.

ERP: a program that listens for event packets on the Ether. Useful for monitoring and gathering statistics.

DOCUMENTATION: ERP.press.

EXECUTIVE: the Alto command processing subsystem, the intermediary by which users generally invoke other subsystems and perform several operations on the Alto file system. Normally invoked by the boot operation.

DOCUMENTATION: Executive.tty or Subsystems.press.

FIND: a subsystem to search one or more text files for a user supplied string at very high speed and then display each line containing an occurrence of the pattern on request.

DOCUMENTATION: FIND.tty or Subsystems.press.

FRED: a part of the Font Creation System, it is used to create and/or edit "splines" (i.e. outlines) of characters.

DOCUMENTATION: [Maxc]KGR-DOCS>Fred.ears.

FTP: a file transfer program to store and retrieve files between an Alto and another Alto, Maxc, or File Server. It also supports a Telnet connection that is similar to CHAT in purpose and operation.
DOCUMENTATION: FTP.tty or Subsystems.press.

GOBBLE: a part of the Design Automation System that generates a wirelist and routing information for a single board given one or more node list files generated by ANALYZE.
DOCUMENTATION: [Maxc]SIL>SilManual.press.

GYPSY: a modeless text editor using both keyset and mouse, and having a "filing cabinet" interface which provides some file management facilities beyond the normal Alto filing system. Used in applications where limited formatting facilities are required such as programming.
DOCUMENTATION: Under development.

HARDCOPY: a wonderful program that retrieves and prints files from anywhere, your Alto, Maxc, a File Server, etc.
DOCUMENTATION: HardCopy.tty.

ICARUS2: a part of the ICARUS2 System, it is an interactive program for actually laying out printed circuits and manipulating the resulting files.
DOCUMENTATION: [Maxc]KICARUS>Icarus2doc.press, ICtools.press.

IF: a program allowing conditional execution of executive commands. Useful in conjunction with DO files and CM files.
DOCUMENTATION: IF.press.

IFD2: a part of the ICARUS2 System that turns an ICARUS2 file into human readable form, describing each symbol and its contents.
DOCUMENTATION: [Maxc]KICARUS>IFD2doc.press.

IFS: the Ivy File System server that provides one end of the file transfer facility and maintains the files and directories on Trident T-80/T-300 disks.
DOCUMENTATION: [Maxc]KIFS>IFSdocuments.press.

IFSSCAVENGER: a subsystem to check and correct Trident T-80/T-300 diskpacks from the Ivy and Trident File Systems.
DOCUMENTATION: [Maxc]KIFS>ScavOp.press.

INSTALLSWAT: an installation program to install the SWAT debugging system on your disk.
DOCUMENTATION: None.

KAL: a kalidescope program that you Etherboot.
DOCUMENTATION: None.

KEYTEST: a diagnostic program that displays the Alto keyboard, keyset and mouse. The depressing of any key(s) is reflected by inverting (white to black) the display of that key on the screen. If the keyboard displayed doesn't match the one you are using, move the cursor to the bottom of the display and hit any mouse button.
DOCUMENTATION: none.

LAUREL: a Maxc MSG compatible, display-based, message system that runs on your Alto.
DOCUMENTATION: [Maxc]Laurel>Laurel.press and a system tutorial retrieved with Laurel software.

LISTSYMS: a programming aid to convert a .Syms file (produced by BLDR) to a useful, human readable form.
DOCUMENTATION: ListSyms.tty or Subsystems.press.

LOGICPROM: Superseded by PROM.

MADTEST: a bootfile diagnostic that runs tests on an Alto's RAM, ALU, and emulator.
DOCUMENTATION: Alto User's Primer.

MAILCHECK: a simple subsystem that checks for mail at some other host (e.g. Maxc) via the Ethernet.

DOCUMENTATION: MailCheck.tty or Subsystems.press.

MARKUP: an illustrator used to add pictures consisting of lines, areas, mouse tracks and text captions to formatted documents, i.e. Press files. It may also be used to simply display press files.
DOCUMENTATION: Alto User's Handbook.

MENUEDIT: a program that edits menus for the BCPL menu package.

DOCUMENTATION: Menu.press.

MIKE: a part of the ICARUS2 System that transforms ICARUS2 files into a form suitable for the Mann 3000 pattern generator.

DOCUMENTATION: [Maxc]ICARUS>MikeUserDoc.press, MikeDoc.press.

MOVETOKEYS: Obsolete, non-functional.

MU: the Alto Microcode assembler.

DOCUMENTATION: MU.tty or Subsystems.press.

NEPTUNE: a small, fast DDS-like program for manipulating your Alto directory.

DOCUMENTATION: Neptune.press.

NETWORK EXECUTIVE: the executive obtained by booting from the Gateway over the Ethernet that provides a convenient way to call "bootfiles" such as FTP or COPYDISK.

DOCUMENTATION: NetExec.tty.

NPGR: Obsolete part of the Sil system.

NPPR: Obsolete part of the Sil system.

OEDIT: a subsystem for displaying and modifying Alto files in octal. Up to four files may be simultaneously viewed while one of them may be modified.

DOCUMENTATION: Oedit.tty or Subsystems.press.

ORBITTEST: the ORBIT interface diagnostic.

DOCUMENTATION: [Ivy]Spruce>ORBITtest.press.

PACKMU: a program to convert the output of MU (an MB file) to a "packed RAM image" which is easy to load into the RAM using RPRAM.

DOCUMENTATION: PackMU.tty or Subsystems.press.

PEEK: a program which listens to the Ethernet for PeekReports and EventReports. It can also serve as a bootserver and Ethernet Echo server for use with EDP.

DOCUMENTATION: DMT.tty.

PEEKPUP: a small subsystem enabling one to peek at Pups going to and from a particular Ethernet host; a debugging aid for new Pup software.

DOCUMENTATION: PeekPup.tty or Subsystems.press.

PEEKSUM: a subsystem that summarizes the error reports sent to PEEK by DMT.
DOCUMENTATION: DMT.tty.

PREPRESS: a part of the Font Creation System that takes "spline character definitions", usually created by FRED, and generates scan-converted characters, spline and character dictionaries, readable listings describing the dictionary's content, and a "widths" file for use by text formating programs.

DOCUMENTATION: [Maxc]GKR-DOCS>PrePress.press.

PRESS: a subsystem to print full press files on press printers.
DOCUMENTATION: [Maxc]GKR-DOCS>PressOps.press.

PRESSEDIT: a program to combine Press files together, convert Ears files (generated by Pub and Bravo) to Press format, selecting certain pages from a Press or Ears file, or to add extra fonts. The output is a Press file.

DOCUMENTATION: PressEdit.tty or Subsystems.press.

PROM: a subsystem to edit microcode, drive the Alto PROM blower and verify PROMs.
DOCUMENTATION: PROM.bravo.

PROMDIAG: Superseded by PROM.

PROOFREADER: an interim English text proofreader that produces an output file listing the questionably-spelled words.
DOCUMENTATION: ProofReader.tty.

PUPTEST: a PUP protocol and network integrity test program.
DOCUMENTATION: None.

PUT: a program for transferring files between the disks of a dual-drive Alto. Its function is also performed by the more comprehensive NEPTUNE.
DOCUMENTATION: Internal to the program.

QED: an in-core line editor used primarily for programming. The file is limited to about 1500 lines of BCPL.
DOCUMENTATION: QED.tty or Subsystems.press.

RAMLOAD: a microcode loader that uses the output of the microcode assembler, MU.
DOCUMENTATION: RamLoad.tty or Subsystems.press.

RAMTIMING: a bootfile diagnostic program to test the Alto RAM. Its function is also performed by the more comprehensive MADTEST.
DOCUMENTATION: None.

READPRESS: reads Press files and displays a text-listing of the entity commands, DL strings, etc.
DOCUMENTATION: Subsystems.press.

RENAME: Obsolete replacement for the Executive's Rename command.

RPRAM: a microcode loader that loads a packed RAM image (generated by PACKMU) into the RAM after checking the constant memory.
DOCUMENTATION: PackMU.tty or Subsystems.tty.

SCAVENGER: a subsystem for checking and correcting Alto disk packs.
DOCUMENTATION: Scavenger.tty or Subsystems.press.

SETTIME: Obsolete.

ShowAIS: a bootfile that halftones and displays 8 bit/pixel AIS files stored on a remote file server. (Use of this program significantly loads the file server, limiting overall performance. Do not use it frivolously.)

DOCUMENTATION: [Ivy]KMaleson>ShowAIS.bravo.

SIGMA: a subsystem to transfer arbitrary files between an Alto and a SIGMA 3 over the Ethernet. DOCUMENTATION: "Ethernet Software for Data Transfer between the SIGMA 3 and an ALTO", a Xerox Internal Report, Accession No. X7704459.

SIL: a part of the Design Automation System, it is an illustrator for the creation of logic and line diagrams. The output can generate Press files or be processed by ANALYZE for circuit design. DOCUMENTATION: [Maxc]SIL>SilManual.press.

SORT: a very small subsystem which will sort files containing less than 1000 entries delimited by a carriage return.

DOCUMENTATION: Subsystems.press.

SPRUCE: a printer server that utilizes the ORBIT buffer to drive Press printers, e.g. Dover and Sequoia.

DOCUMENTATION: [Maxc]Spruce>SpruceManual.press.

SWAT: a emulator-level code debugger with BCPL oriented features used with the Alto operating system.

DOCUMENTATION: Swat.tty or Subsystems.press.

SYS.BOOT: the operating system boot file on the Alto disk.

DOCUMENTATION: None.

TFU! a file utility used to initialize a Trident pack with a virgin file system and to perform various file copying, deleting, directory listing operations. This is not a part of the Ivy System, rather it initializes and maintains packs operated on by the TFS package.

DOCUMENTATION: TFS.tty or Subsystems.press.

TRANSFILE: a part of the ICARUS2 System that translates the intermediate files generated by Mike into human-readable form. The files it produces are fully instantiated.

DOCUMENTATION: [Maxc]ICARUS>TransfileDoc.press.

TRIEX: a Trident diagnostic used to debug and exercise Trident disk drives.

DOCUMENTATION: Self-contained.

TYPE: a functional replacement to the Executive supplied "type.~" that displays a larger page, suppresses Bravo trailer information, can skip forward and backward, etc.

DOCUMENTATION: TYPE.tty.

UGH: an in-core text editor utilizing both mouse and keyset. While some formating facilities are available, it is used primarily for programming.

DOCUMENTATION: UGH.tty.

VIEWDATA: a subsystem to display on the Alto screen three-dimensional data stored as a two-dimensional array of single-word values.

DOCUMENTATION: ViewData.tty.

VIEWIC: a part of the ICARUS2 System that displays the data created by Mike on the Alto screen, simulating the actions of the pattern generator.

DOCUMENTATION: [Maxc]KICARUS>ViewicDoc.press.

VPRINT: a subsystem to output text files such as .TTY, UGH, BRAVO, or GYPSY, to a Versatec printer.

DOCUMENTATION: Under development.

FUNCTIONAL CROSS REFERENCE

The following list of Alto subsystems is organized according to the general function they perform. Because many subsystems perform more than one function or a function may be thought of in a variety of ways, an item may be listed more than once.

The major functional headings are:

Document Creation	Hardware Design	Messages
Files	Hardware Diagnostics	Printing
Font Creation	Hardware Drivers	Programming
	Recovering	

DOCUMENT CREATION

EDITORS

TEXT

- BRAVO: Rich in formatting features.
- GYPSY: Features to handle groups of files (e.g. chapters or modules).
- PROOFREADER: Produces an output file of questionably spelled words.
- QED: A line editor.
- UGH: An in-core editor that uses the keyset for command input.

GRAPHIC

- DRAW: Pictures composed of lines, curves, text and smoothed mouse tracks.
- FREID: A Spline editor for font work.
- MARKUP: Dot pictures of lines, areas, text, and mouse tracks.
- SIL: For creating diagrams composed of lines with text captions.

IMAGES

- AIS: Image manipulation, printing, Press file creation.

PAGE MAKEUP

- MARKUP: Create new or move pre-existing diagrams along side existing text.

MERGING

- AIS: Merges bitmap files (e.g. PRESS output and AIS files).
- EMPRESS: Append press files to personalized coversheets.
- PRESSEdit: Generate Press files from pages of other Press files.

PRINTING: see PRINTING below.

FILES

DISPLAY

- AISshow: AIS files.
- OEDIT: Alto files in octal.
- MARKUP: Press files.

PRESS: Press files.
READPRESS: Press file internals.
ShowAIS: 8 bit/pixel AIS files from remote file servers.
TYPE: Text files.
VIEWDATA: Three-dimensional information stored as a matrix of values.

TRANSFER

COPYDISK: Copies whole disks, Diablo Models 31/44 and Trident T-80/T-300.
FTP: Copies a file between Altos, Alto-File Server, and Alto-Maxc.
GYPSY: Transmits its files to a Communicating 800 ETS.
NEPTUNE: Copies files between disks of a dual drive Alto.
PUT: Copies files between disks of a dual drive Alto.
SIGMA: Copies files between Alto and SIGMA 3.
TFU: Copies files between Trident drives.

ALTO FILE SYSTEM

CALLFTP: Very small version of FTP.
CLEANDIR: Garbage collect disk directory.
CREATEFILE: Adds new file of specified size on consecutive pages, if possible.
CRUMBLE: Compresses and optionally encrypts a file.
DDS: Large directory manipulation subsystem.
EXECUTIVE: Delete files, list directory.
FIND: Locates and displays lines containing a specified text string.
FTP: Transfers files between Alto and Alto, a File Server, or Maxc.
NEPTUNE: Small directory manipulation subsystem. Deletes, renames, and copies files between disks of a dual drive Alto.
OEDIT: Display and modify file in octal.
PUT: Deletes, renames, and copies files between disks of a dual drive Alto.
SCAVENGER: Checks and corrects the disk.
SIGMA: Transfers files between an Alto and SIGMA 3.
SORT: Sorts up to 1000 items delimited by carriage returns.
TYPE: Displays contents of text files.

FILE SERVER SYSTEM

IFS: The server.
IFSSCAVENGER: Checks and corrects Trident T-80/T-300 disks.
TFU: Initialize directory and verify disk.

TRIDENT FILE SYSTEM

IFSSCAVENGER: Checks and corrects Trident T-80/T-300 disks.
TFU: Initialize and list directory, verify disk, copy and delete files.

RECOVERY

IFSSCAVENGER: Check and correct disks of File Server and Trident file systems.
SCAVENGER: Check and correct disk of Alto file system.

FONT CREATION

SPLINES

DRAW: Create splines using mouse or knots.

FRED: Create and edit splines, create font files.

BITMAPS

PREPRESS: Scales and rotates splines, converts to and edits bitmaps.

DEVICE FORMATS

PREPRESS: Creates printer and display fonts from bitmaps.

HARDWARE DESIGN

CIRCUIT BOARD

SIL: Create and edit logic diagrams.

ANALYZE: Converts SIL drawing for GOBBLE, generates SIL of unassigned pins.

GOBBLE: Generates wirelist and routing information.

BUILD: Aids data management aspects, keeping data files current.

INTEGRATED CIRCUIT

ICARUS2: Layout integrated circuits.

IFD2: Generates human readable form of ICARUS2 files.

MIKE: Transforms ICARUS2 file to form for Mann 3000 pattern generator.

TRANSFILE: Generates human readable form of MIKE file.

VIEWIC: Displays MIKE output.

ICGERB: Generates Gerber photoplotter output from ICARUS files.

HARDWARE DIAGNOSTICS

USER

CRTTEST: Displays a rectangular grid. (.boot/.run files).

DMT: Memory diagnostic that transmits results to PEEK. (Alto/server boot files).

EDP: Ethernet interface diagnostic.

KEYTEST: Keyboard diagnostic. (.boot/.run files).

MADTEST: Diagnostic for RAM, ALU, and emulator. (.boot/.run files).

TRIEX: A Trident T-80/T-300 diagnostic.

INSTALLATION

DISKTEST: Bootfile diagnostic for the Diablo Model 31.

ORBITTEST: An ORBIT Interface diagnostic.

PEEK—Collects PeekReport/EventReport packets on <filename>.

PEEKSUM—Summarizes DMT error reports collected by PEEK.

PUPTEST—Ascertains status of network servers.

HARDWARE DRIVERS—See the Alto Hardware Catalog under appropriate device.

MESSAGES**SENDING**

CHAT—Accesses SNDMSG on Maxc.
LAUREL—Display-based, MSG compatible, running on Alto.

RECEIVING

MAILCHECK—Interrogates Maxc for new mail.
CHAT—Accesses MSG on Maxc to retrieve mail.
LAUREL—Display-based, MSG compatible, running on Alto.

PRINTING**REMOTE****REFORMATING**

PREPRESS—Generates a Press file from .tty and many .press files.

PRESS

BRAVO—Sends the workfile to Spruce.
EMPRESS—Sends the specified Press and/or Text files to Spruce.
HARDCOPY—Prints any file any place.
GYPSY—Sends the workfile to Spruce.
SPRUCE—A Press server for Dover/Sequoia/Pimlico.

LOCAL

BRAVO—Prints the workfile on the attached Diablo HyType.
DPRINT—Prints text files on the attached Diablo HyType.
GYPSY—Prints the workfile on the attached Diablo Hytype.
PRESS—Prints Press files on slot and Versatec printers.
VPRINT—Prints text files on Versatec printers.

PROGRAMMING

EDITORS—See DOCUMENT-EDITORS-TEXT above.

ASM/BCPL

ASM—Alto machine language assembler.
BCPL—BCPL compiler.
BLDR—Loader for ASM and BCPL relocatable files.
BUILDBOOT—Generates a type B bootfile.
SWAT—an emulator level, BCPL oriented debugger.
INSTALLSWAT—Installs SWAT on a disk.
LISTSYMS—Converts .SYMS files to human readable form.

MU

MU—Microcode assembler.
RAMLOAD—Loads RAM with MU output.
PACKMU—Converts MU output for RPRAM.
RPRAM—Loads RAM with RPRAM output.

DEBUGGING AIDS

AISdump—Writes out pixel values in decimal.
CONDENSE—Recovers display bitmap from SWAT or SWATEE file.
SWAT—An emulator level, BCPL oriented debugger.
LISTSYMS—Converts SYMS files to human readable form.
BTREETEST—A B-tree dictionary maintenance program.
PEEKPUP—Peeks at PUPs going to and from a specific Ethernet address.
PUPTEST—Interacts with new subsystems that use the PUP protocol.
READPRESS—Displays entities within a Press file.

RECOVERING

DISK FAILURES—See FILES-RECOVERY above.

SUBSYSTEMS FAILURES

BRAVOBUG—Recovers BRAVO files to point of failure.
CONDENSE—Recovers display bitmap from SWAT or SWATEE file.