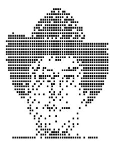
Face the Nation

Rob Pike *David L. Presotto*
research!rob *research!presotto*



AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Digitized images of the faces of members of AT&T Bell Laboratories' Computing Science Research Center, and of many of their electronic mail correspondents, are stored in a network 'face server' accessible from the Eighth Edition machines at Murray Hill. The faces decorate line printer banners, announce the arrival of mail, and perform a variety of other Information Age services. The face server is implemented as a variant of the network file system, so the images are stored on one machine but appear to be regular files in the file systems of all the machines on the network. This method of presenting the information is attractive for databases with inherent structure, as it permits standard Unix system utilities to access, manipulate and maintain the database.

Spurred on by initial successes with some experiments with famous people (E.W. Dijkstra and P.J. Weinberger), we last year undertook to digitize the faces of all people in our research center and of those with whom we communicate. The result of this effort was about thirty megabytes of disk space, which soon began to spread to other computers that needed access to the faces for announcement of mail, banners on printed output, and so on. This paper presents the design, implementation and use of a network 'face server' that eliminates the need for duplication. The faces are stored in one place that allows all machines and all users to access this treasury of useful information.

Polaroid 4" by 5" black and white photographs of each participant, lit fairly flat with a white background, were digitized by a video frame grabber into 512×512 byte (8 bits per pixel) grey-scale images. Each image was then reduced interactively to a 48×48 bit (1 bit per pixel) black-and-white 'ikon.' The reduction program uses the Floyd-Steinberg algorithm [Newm79]; the interactive activity is setting the upper and lower thresholds to adjust brightness and contrast, then selecting the best-looking image. Although the small pictures are surprisingly recognizable, the quality of an ikon depends critically on the parameters. It seems to need human judgement to choose the best digitization. After the first day's photography session we had over a hundred faces recorded, and it became obvious we needed a central place to store, maintain and administer the face files — a network face server.

The faces were originally stored as ordinary files in a single directory, so the tools to access them were standard programs such as `cat`(1). But instead of a simple list of files, the structure we wanted was hierarchical, with users of a particular machine, or machines in an organization, grouped together. The obvious way to construct this hierarchy was in the file system, with links to associate faces (files) with the many machines (directories) they inhabit. To make the file system available everywhere, the face server simulates a Unix† file system that can be mounted on the client machine using the Eighth Edition network

† Unix is a trademark of AT&T Bell Laboratories, for what it's worth.

file system protocols. Faces are named by ordinary file names, so no special software is required to use them. (Pike and Weinberger [Pike85] discuss other issues related to this naming scheme.) The details of the file system are different, however, to match the structure of the database and its typical uses.

By contrast, the traditional model of a network server is fairly complicated and different from our usual methods of accessing data. A typical server involves operations such as ‘attach protocols’ and ‘transactions’ instead of system calls such as `open`(2) and `read`(2). Inventing new methods of accessing data requires inventing new software to use the methods, but the standard system calls provide all the functionality needed.

Our face server has two file formats. The small ikons are stored as ASCII hexadecimal strings as they would appear in a C declaration, while high-resolution grey-scale images are stored as binary files for storage efficiency. The file system appears as a directory, conventionally `/n/face`, with constituent files named, for example, `/n/face/research/pjw/48x48x1`. The first subdirectory contains machine names, the second users on those machines, and the third the actual files, named by their resolution.

Our first implementation used the regular Eighth Edition network file system [Wein84], which allows a remote system’s root directory to be logically mounted on a node in the local file system. A simple early implementation of the face server mounted the root of the server system on the file system of each other machine (the client). This used only existing software but caused problems. First, access protection was not implemented in a manner appropriate to a network service. The network file system made visible all the server machine’s files, not just the faces. Also, because of the way the network file system implements protections, anyone wanting to use the faces needed an account on the server machine. These details left us with an uncomfortable choice: either to provide to users on client systems more access than they needed, or to deny certain users or systems access to the face server. Second, resources on the server machine were strained by the implementation. The remote file system requires the server system to maintain an open network connection and a server process for each of its client systems. Since we have dozens of systems, this caused the server system to run out of communications channels, process slots, or swap space. Finally, the desired multiply-connected view of the data was clumsy and inefficient when built with traditional file system techniques: the large number of symbolic links were expensive in both disk space and CPU time to access.

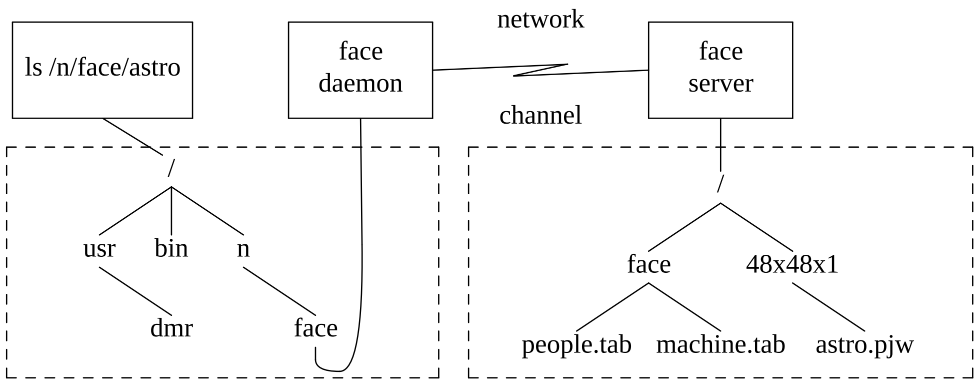
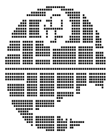
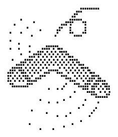


Figure 1. Structure of the face server.

The current implementation avoids these problems by dividing the work of the server between two different processes while leaving the flow of data through the kernel exactly the same as for a regular network file system (see Figure 1). A daemon process runs on each client system and a single server process runs on the server system. Each client daemon maintains the state information concerning open face files for its system. These daemons translate file system primitives to remote procedure calls to the server process across the network. Because of the low duty cycle on a connection and the relative scarcity of channels on the service machine, the link drops automatically if it is not used for a few minutes. The server is stateless in the sense that all incoming requests have full information; the server does not remember information about calls. This simplifies the job of creating and shutting down connections, and results in greater reliability and resilience to failures (such as system crashes) than for the regular network file system. The file access problems are solved by making only face files visible through the face server. This makes it possible to apply special protection methods to face files without affecting access to other files on the server system.

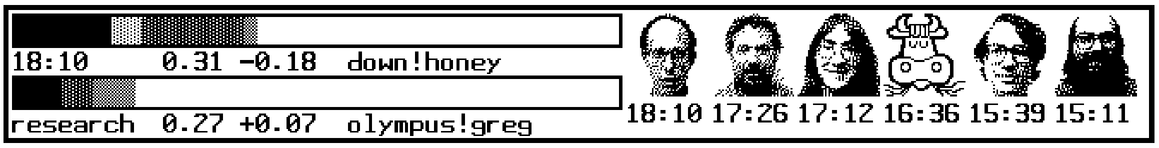
The server reads a pair of files (one for machine names, one for people names) that describe the correspondence between machine/person pairs and regular files on disk, and builds an in-core tree structure corresponding to the face file system's directory tree. The leaves of the tree are pointers to the actual files on disk, but the directories are kept in core so directories can be 'hard linked' together (this cannot be done on disk because the link bits in a directory i-node are usurped for directory tree consistency checking); machines related by organization (e.g. mit-eddie and MIT-MC) are linked to a directory that names the organization (MIT). Maintenance consists of updating the description files to reflect the correspondence between the real world and the availability of images.

The face server provides a simple database of faces; there is also a conventional protocol to convert a machine/person pair to a face. The directory for an organization may contain an 'unknown' face to be used when the appropriate user's face is not available, and there is an artificial organization 'misc.' to store ikons for generic users such as root and uucp. The following faces are those retrieved for `research!pjw` (standard available face), `lucasfilm!george` (the unknown face for lucasfilm, `lucasfilm/unknown`) and `decvax!uucp` (the general uucp ikon, `misc./uucp`):

On rare occasions, such as when its host machine has just had preventive maintenance, the face server may be down. For such times, some facial programs keep internally a blank face not stored in the face server, so some face is always available:

The main client of the face server is a program called `vismon` that continually monitors the CPU load on systems on the network and reports the arrival of mail. The sender of each message is converted to a face and displayed in `vismon`'s window (see Figure 2). The result of an afternoon's mail is a police lineup of faces, perhaps with a hand-drawn ruminant.



```
18:10     0.31 -0.18  down!honey
research  0.27 +0.07  olympus!greg
18:10 17:26 17:12 16:36 15:39 15:11
```

Figure 2. A `vismon` window, showing a collection of faces and the system activity on a couple of machines.

Although small, the images are remarkably good likenesses of their subjects, and because the picture of a given person is the same from day to day, it comes to represent the person strongly; the word ikon is particularly appropriate. (*Chambers 20th Century dictionary* defines ikon as, ''a symbol, representation: anybody or anything uncritically admired.'') At time of writing we have 263 faces in our database, but only about 60 in our own research center. The other two hundred are people outside our group, some even from other continents, and therefore less familiar. Nonetheless, the face images make their owners more a part of the local community; a face seen only once a month is more recognizable than three-letter initials seen daily.

The lessons from the face server involve the method of presentation of the database: as a set of regular Unix files. By providing conventional names for the faces, ordinary Unix tools can be used as database utilities. Less obvious, the behavior of files is so well known that there was very little work to do in building the service; the phrase ‘‘file system’’ determines most of the details. We resisted the temptation to extend the semantics of the files by, for example, creating files of particular resolution on demand or coding the name of the file to determine ASCII or binary format; instead, the face server provides regular Unix files, building on their behavior rather than changing it. The implementation was made simpler by the existence of a working network file system and the Eighth Edition IPC mechanisms. Now that we have one database working, others may appear. One possibility is a digital font server implemented as a file system.

Acknowledgements

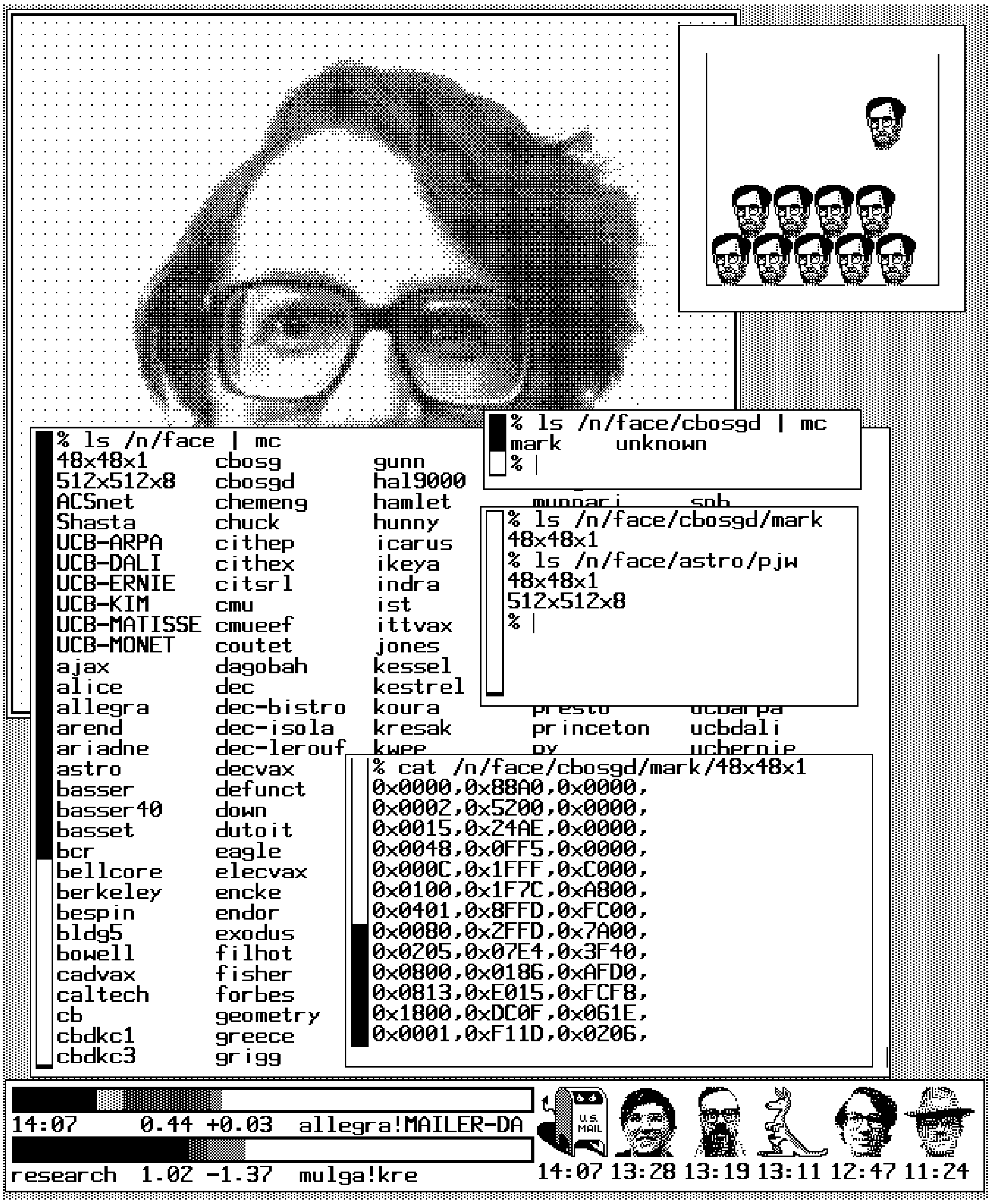
Luca Cardelli drew the first few ikons by hand, ran the digitizing hardware and software for the first hundred or so faces, and deserves credit for putting faces on our minds. P.J. Weinberger provided the network file system and an inspiring visage.

References

[Newm79] W.M. Newman and R.F. Sproull, *Principles of Interactive Computer Graphics*, p. 226, McGraw-Hill, New York 1979.

[Pike85] Rob Pike and P.J. Weinberger, ‘‘The Hideous Name,’’ *Summer 1985 USENIX Conference Proceedings*, Portland, Oregon.

[Wein84] P.J. Weinberger, ‘‘The Version 8 Network File System,’’ *Summer 1984 USENIX Conference Proceedings*, Salt Lake City, Utah.



```
% ls /n/face | mc
48x48x1         cbosg           gunn
512x512x8       cbosgd          hal9000
ACSnet          chemeng         hamlet          munnari         snb
Shasta          chuck           hunny
UCB-ARPA        cithep          icarus
UCB-DALI        cithex          ikeya
UCB-ERNIE       citsrl          indra
UCB-KIM         cmu             ist
UCB-MATISSE     cmueef          ittvax
UCB-MONET       coutet          jones
ajax            dagobah         kessel
alice           dec             kestrel
allegra         dec-bistro      koura           presto          ucbarpa
arend           dec-isola       kresak          princeton       ucbdali
ariadne         dec-lerouf      kwee            py              ucbernie
astro           decvax
basser          defunct
basser40        down
basset          dutoit
bcr             eagle
bellcore        elecvax
berkeley        encke
bespin          endor
bldg5           exodus
bowell          filhot
cadvax          fisher
caltech         forbes
cb              geometry
cbdkc1          greece
cbdkc3          grigg
```

```
% ls /n/face/cbosgd | mc
mark    unknown
% |
```

```
% ls /n/face/cbosgd/mark
48x48x1
% ls /n/face/astro/pjw
48x48x1
512x512x8
% |
```

```
% cat /n/face/cbosgd/mark/48x48x1
0x0000,0x88A0,0x0000,
0x0002,0x5200,0x0000,
0x0015,0x24AE,0x0000,
0x0048,0x0FF5,0x0000,
0x000C,0x1FFF,0xC000,
0x0100,0x1F7C,0xA800,
0x0401,0x8FFD,0xFC00,
0x0080,0x2FFD,0x7A00,
0x0205,0x07E4,0x3F40,
0x0800,0x0186,0xAFD0,
0x0813,0xE015,0xFCF8,
0x1800,0xDC0F,0x061E,
0x0001,0xF11D,0x0206,
```

```
14:07     0.44 +0.03  allegra!MAILER-DA
research  1.02 -1.37  mulga!kre
```

Figure 3. A Teletype DMD-5620 screen displaying the result of an afternoon’s activity to create a Figure 3 showing the face server being used. The top two programs show a dithered 512×512×8 face and an early experiment with hand-drawn faces. The central group of programs illustrate the structure of the face file system. The bottom program is a `vismon`, described in the text.