

Very first version of Git in Cit

```

contents
1 README
2 Makefile
3 cache.h
4 read-cache.c
5 update-cache.c
6 init-db.c
7 commit-tree.c
8 read-tree.c
9 write-tree.c
10 cat-file.c
11 show-dif.c

```

```

contents
1 if (stat(ce->name, &st) < 0) {
2   perror("stat(%s: %s)", ce->name, strerror(errno));
3   continue;
4 }
5 if (changelog = match_start(ce, &st)) {
6   printf("%s: %s", ok"\n", ce->name);
7   for (n = 0; n < 20; n++)
8     printf("%02x", ce->shead[n]);
9   printf("\n");
10  free(new);
11  new = read_shallow_file(ce->shad, type, &size);
12  show_differences(ce, &st, new, size);
13  free(new);
14  return 0;
15 }
16 void *new;
17 char *type[2];
18 unsigned int mode;
19 struct cache_entry *ce = active_cache[i];
20 int n, changelog;
21 if (streq(cache_entry->ce->name, "refs/heads/master"))
22   printf("%s: %s", ok"\n", ce->name);
23   continue;
24 else if (streq(cache_entry->ce->name, "refs/heads/HEAD"))
25   printf("%s: %s", ok"\n", ce->name);
26   continue;
27 else if (streq(cache_entry->ce->name, "refs/heads"))
28   printf("%s: %s", ok"\n", ce->name);
29   continue;
30 else if (streq(cache_entry->ce->name, "refs"))
31   printf("%s: %s", ok"\n", ce->name);
32   continue;
33 else if (streq(cache_entry->ce->name, "refs/heads"))
34   printf("%s: %s", ok"\n", ce->name);
35   continue;
36 else if (streq(cache_entry->ce->name, "refs/heads"))
37   printf("%s: %s", ok"\n", ce->name);
38   continue;
39 else if (streq(cache_entry->ce->name, "refs/heads"))
40   printf("%s: %s", ok"\n", ce->name);
41   continue;
42 else if (streq(cache_entry->ce->name, "refs/heads"))
43   printf("%s: %s", ok"\n", ce->name);
44   continue;
45 else if (streq(cache_entry->ce->name, "refs/heads"))
46   printf("%s: %s", ok"\n", ce->name);
47   continue;
48 else if (streq(cache_entry->ce->name, "refs/heads"))
49   printf("%s: %s", ok"\n", ce->name);
50   continue;
51 else if (streq(cache_entry->ce->name, "refs/heads"))
52   printf("%s: %s", ok"\n", ce->name);
53   continue;
54 else if (streq(cache_entry->ce->name, "refs/heads"))
55   printf("%s: %s", ok"\n", ce->name);
56   continue;
57 else if (streq(cache_entry->ce->name, "refs/heads"))
58   printf("%s: %s", ok"\n", ce->name);
59   continue;
60 else if (streq(cache_entry->ce->name, "refs/heads"))
61   printf("%s: %s", ok"\n", ce->name);
62   continue;
63 else if (streq(cache_entry->ce->name, "refs/heads"))
64   printf("%s: %s", ok"\n", ce->name);
65   continue;
66 else if (streq(cache_entry->ce->name, "refs/heads"))
67   printf("%s: %s", ok"\n", ce->name);
68   continue;
69 else if (streq(cache_entry->ce->name, "refs/heads"))
70   printf("%s: %s", ok"\n", ce->name);
71   continue;
72 printf("%s: %s", ok"\n", ce->name);
73   continue;
74 for (n = 0; n < 20; n++)
75   printf("%02x", ce->shead[n]);
76   printf("\n");
77   free(new);
78   new = read_shallow_file(ce->shad, type, &size);
79   show_differences(ce, &st, new, size);
80   free(new);
81 }
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

1 README

GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager. It doesn't do a whole lot, but what it does_ do is track directory contents efficiently.

There are two object abstractions: the "object database", and the "current directory cache".

The Object Database (SHA1_FILE_DIRECTORY)

The object database is literally just a content-addressable collection of objects. All objects are named by their content, which is approximated by the SHA1 hash of the object itself. Objects may refer to other objects (by referencing their SHA1 hash), and so you can build up a hierarchy of objects.

There are several kinds of objects in the content-addressable collection database. They are all in deflated with zlib, and start off with a tag of their type, and size information about the data. The SHA1 hash is always the hash of the compressed object, not the original one.

In particular, the consistency of an object can always be tested independently of the contents or the type of the object: all objects can be validated by verifying that (a) their hashes match the content of the file and (b) the object successfully inflates to a stream of bytes that forms a sequence of <ascii tag without space> + <space> + <ascii decimal size> + <byte\0> + <binary object data>.

BLOB: A "blob" object is nothing but a binary blob of data, and doesn't refer to anything else. There is no signature or any other verification of the data, so while the object is consistent (it is indexed by its sha1 hash, so the data itself is certainly correct), it has absolutely no other attributes. No name associations, no permissions. It is purely a blob of data (ie normally "file contents").

TREE: The next hierarchical object type is the "tree" object. A tree object is a list of permission/name/blob data, sorted by name. In other words the tree object is uniquely determined by the set contents, and so two separate but identical trees will always share the exact same object.

Again, a "tree" object is just a pure data abstraction: it has no history, no signatures, no verification of validity, except that the contents are again protected by the hash itself. So you can trust the

11 show-diff.c

```
1 #include "cache.h"
2
3 #define MTIME_CHANGED      0x0001
4 #define CTIME_CHANGED      0x0002
5 #define OWNER_CHANGED      0x0004
6 #define MODE_CHANGED       0x0008
7 #define INODE_CHANGED      0x0010
8 #define DATA_CHANGED       0x0020
9
10 static int match_stat(struct cache_entry *ce, struct stat *st)
11 {
12     unsigned int changed = 0;
13
14     if (ce->mtime.sec != (unsigned int)st->st_mtim.tv_sec ||
15         ce->mtime.nsec != (unsigned int)st->st_mtim.tv_nsec)
16         changed |= MTIME_CHANGED;
17
18     if (ce->ctime.sec != (unsigned int)st->st_ctim.tv_sec ||
19         ce->ctime.nsec != (unsigned int)st->st_ctim.tv_nsec)
20         changed |= CTIME_CHANGED;
21
22     if (ce->st_uid != (unsigned int)st->st_uid ||
23         ce->st_gid != (unsigned int)st->st_gid)
24         changed |= OWNER_CHANGED;
25
26     if (ce->st_mode != (unsigned int)st->st_mode)
27         changed |= MODE_CHANGED;
28
29     if (ce->st_dev != (unsigned int)st->st_dev ||
30         ce->st_ino != (unsigned int)st->st_ino)
31         changed |= INODE_CHANGED;
32
33     if (ce->st_size != (unsigned int)st->st_size)
34         changed |= DATA_CHANGED;
35
36     return changed;
37 }
38
39 static void show_differences(struct cache_entry *ce, struct stat *cur,
40                             void *old_contents, unsigned long long old_size)
41 {
42     static char cmd[1000];
43     FILE *f;
44
45     sprintf(cmd, sizeof(cmd), "diff -u - %s", ce->name);
46     f = popen(cmd, "w");
47     fwrite(old_contents, old_size, 1, f);
48     pclose(f);
49 }
50
51 int main(int argc, char **argv)
52 {
53     int entries = read_cache();
54     int i;
55
56     if (entries < 0) {
57         perror("read_cache");
58         exit(1);
59     }
60
61     for (i = 0; i < entries; i++) {
62         struct stat st;
```

In other words, you can easily validate a whole archive by just sending out a single mail that tells the people the name (SHA1 hash) of the top changeset, and digitally sign that email using something like GPG/GPG.

So to introduce some real trust in the system, the only thing you need to do is to digitally sign just one specific note, which includes the name of a top-level changeset. Your digital signature shows others that you trust that changeset, and the immutability of the history of changesets tells others that they can trust the whole history.

Furthermore, since the SHA1 signature of a changeset refers to the SHA1 signatures of the tree it is associated with and the signatures of the parent, a single name of a changeset, with full contents. You can't later fake any step of the way once you have the name of a changeset.

TRUST: The notion of "trust" is really outside the scope of "git", but it's worth noting a few things. First off, since everything is hashed with SHA1, you can trust that an object is intact and has not been modified by external sources. So the same of an object uniquely messes with by external sources. So the same of an object uniquely identifies a known state - just not a state that you may want to trust.

Note on changesets: unlike real SCM's, changesets do not contain rename information or file mode change information. All of that is implied in parents', and describes that makes no sense in this directory manager.

A "changeset" is defined by the tree-object that it results in, the parents do not have to actually have any relationship with the result, the tree is "good" or that the merge transformation makes sense. The strong signatures at all levels, but there is no reason to believe that contents are well-defined and "safe" due to the cryptographic safety.

Comment on what happened. Again, a changeset is not trusted per se: parent changesets (zero, one or more) that led up to that point, and a comment on what happened.

CHANGESET: The "changeset" object is an object that introduces the notion of history into the picture. In contrast to the other objects, it doesn't just describe the physical state of a tree, it describes how we got there, and why.

Side note 2 on trees: since the name of a "blob" depends entirely and actually having to unpack two trees. Just ignore all common parts, and effectively having to unpack two trees. In other words, just ignore all common parts, and your diff will look right. In other words, just ignore all common parts, and where "n" is the size of the difference, rather than the size of the involved), you can see tree random names or permissions changes by noticing that the blob stayed the same. However, names with data changes need a smarter "diff" implementation.

Side note on trees: since a "tree" object is a sorted list of contents of a tree, the same way you can trust the contents - came from,

but you don't know those contents - came.

```

1 #include "cache.h"
2 cat-file.c
3 int main(int argc, char **argv)
4 {
5     unsigned char sha1[20];
6     char type[20];
7     void *buf;
8     unsigned long size;
9     char template[] = "temp-git-file-XXXXXX";
10    int fd;
11    if (argc != 2 || !get_sha1_hex(argv[1], sha1))
12        usage("cat-file: cat-filename <sha1>:");
13    buf = read_sha1_file(sha1, type, &size);
14    if (fd = mkstemp(template));
15        exit(1);
16    if (write(fd, buf, size) != size)
17        strcpy(type, "bad");
18    if (fd > 0)
19        usage("unable to create template");
20    if (fclose(fd));
21        perror("fd: %s", strerror(errno));
22    if (fstat(fd, &st));
23    printf("%s: %s", type, template, type);

```

In particular, you can also have a separate archive of "trust points" or tags, which document your (and other peoples) trust. You may, of course, archive these "certificates of trust" using "git" itself, but it's not something "git" does for you.

Another way of saying the same thing: "git" itself only handles content integrity, the trust has to come from outside.

Current Directory Cache ("dircache/index")

The "current directory cache" is a simple binary file, which contains an efficient representation of a virtual directory content at some random time. It does so by a simple array that associates a set of names, dates, permissions and content (aka "blob") objects together. The cache is always kept ordered by name, and names are unique at any point in time, but the cache has no long-term meaning, and can be partially updated at any time.

In particular, the "current directory cache" certainly does not need to be consistent with the current directory contents, but it has two very important attributes:

- (a) it can re-generate the full state it caches (not just the directory structure: through the "blob" object it can regenerate the data too)

As a special case, there is a clear and unambiguous one-way mapping from a current directory cache to a "tree object", which can be efficiently created from just the current directory cache without actually looking at any other data. So a directory cache at any one time uniquely specifies one and only one "tree" object (but has additional data to make it easy to match up that tree object with what has happened in the directory)

and

- (b) it has efficient methods for finding inconsistencies between that cached state ("tree object waiting to be instantiated") and the current state.

Those are the two ONLY things that the directory cache does. It's a cache, and the normal operation is to re-generate it completely from a known tree object, or update/compare it with a live tree that is being developed. If you blow the directory cache away entirely, you haven't lost any information as long as you have the name of the tree that it described.

(But directory caches can also have real information in them: in particular, they can have the representation of an intermediate tree that has not yet been instantiated. So they do have meaning and usage outside of caching - in one sense you can think of the current directory cache as being the "work in progress" towards a tree commit).

```
56     i = prepend_integer(buffer, offset - ORIG_OFFSET, ORIG_OFFSET);
57     i -= 5;
58     memcpy(buffer+i, "tree ", 5);
59
60     buffer += i;
61     offset -= i;
62
63
64     write_sha1_file(buffer, offset);
65
66 }
```

2 Makefile

3 cache.h

```
1 #ifndef CACHE_H
2 #define CACHE_H
3
4 #include <stdio.h>
5 #include <sys/stat.h>
6 #include <fcntl.h>
7 #include <stddef.h>
8 #include <stdlib.h>
9 #include <stdarg.h>
10 #include <errno.h>
11 #include <sys/mman.h>
12
13 #include <openssl/sha.h>
14 #include <zlib.h>
15
16 /*
17 * Basic data structures for the directory cache
18 *
19 * NOTE NOTE NOTE! This is all in the native CPU byte format. It's
20 * not even trying to be portable. It's trying to be efficient. It's
21 * just a cache, after all.
22 */
23
24 #define CACHE_SIGNATURE 0x44495243 /* "DIRC" */
25 struct cache_header {
26     unsigned int signature;
27     unsigned int version;
28     unsigned int entries;
29     unsigned char sha1[20];
30 };
31
32 /*
33 * The "cache_time" is just the low 32 bits of the
34 * time. It doesn't matter if it overflows - we only
35 * check it for equality in the 32 bits we save.
36 */
37 struct cache_time {
38     unsigned int sec;
39     unsigned int nsec;
40 };
41
42 /*
43 * dev/inode/uid/gid/size are also just tracked to the low 32 bits
44 * Again - this is just a (very strong in practice) heuristic that
45 * the inode hasn't changed.
46 */
47 struct cache_entry {
48     struct cache_time ctime;
49     struct cache_time mtime;
50     unsigned int st_dev;
51     unsigned int st_ino;
52     unsigned int st_mode;
53     unsigned int st_uid;
54     unsigned int st_gid;
55     unsigned int st_size;
```

8 read-tree.c

```
1 #include "cache.h"
2
3 static int unpack(unsigned char *sha1)
4 {
5     void *buffer;
6     unsigned long size;
7     char type[20];
8
9     buffer = read_sha1_file(sha1, type, &size);
10    if (!buffer)
11        usage("unable to read sha1 file");
12    if (strcmp(type, "tree"))
13        usage("expected a 'tree' node");
14    while (size) {
15        int len = strlen(buffer)+1;
16        unsigned char *sha1 = buffer + len;
17        char *path = strchr(buffer, ' ') + 1;
18        unsigned int mode;
19        if (size < len + 20 || sscanf(buffer, "%o", &mode) != 1)
20            usage("corrupt 'tree' file");
21        buffer = sha1 + 20;
22        size -= len + 20;
23        printf("%o %s (%s)\n", mode, path, sha1_to_hex(sha1));
24    }
25    return 0;
26 }
27
28 int main(int argc, char **argv)
29 {
30     int fd;
31     unsigned char sha1[20];
32
33     if (argc != 2)
34         usage("read-tree <key>");
35     if (get_sha1_hex(argv[1], sha1) < 0)
36         usage("read-tree <key>");
37     sha1_file_directory = getenv(DB_ENVIRONMENT);
38     if (!sha1_file_directory)
39         sha1_file_directory = DEFAULT_DB_ENVIRONMENT;
40     if (unpack(sha1) < 0)
41         usage("unpack failed");
42     return 0;
43 }
```


4 read-cache.c

```

1 #include "cache.h"
2
3 const char *sha1_file_directory = NULL;
4 struct cache_entry **active_cache = NULL;
5 unsigned int active_nr = 0, active_alloc = 0;
6
7 void usage(const char *err)
8 {
9     fprintf(stderr, "read-tree: %s\n", err);
10    exit(1);
11 }
12
13 static unsigned hexval(char c)
14 {
15     if (c >= '0' && c <= '9')
16         return c - '0';
17     if (c >= 'a' && c <= 'f')
18         return c - 'a' + 10;
19     if (c >= 'A' && c <= 'F')
20         return c - 'A' + 10;
21     return ~0;
22 }
23
24 int get_sha1_hex(char *hex, unsigned char *sha1)
25 {
26     int i;
27     for (i = 0; i < 20; i++) {
28         unsigned int val = (hexval(hex[0]) << 4) | hexval(hex[1]);
29         if (val & ~0xf)
30             return -1;
31         *sha1++ = val;
32         hex += 2;
33     }
34     return 0;
35 }
36
37 char * sha1_to_hex(unsigned char *sha1)
38 {
39     static char buffer[50];
40     static const char hex[] = "0123456789abcdef";
41     char *buf = buffer;
42     int i;
43
44     for (i = 0; i < 20; i++) {
45         unsigned int val = *sha1++;
46         *buf++ = hex[val >> 4];
47         *buf++ = hex[val & 0xf];
48     }
49     return buffer;
50 }
51
52 /* NOTE! This returns a statically allocated buffer, so you have to be
53 * careful about using it. Do a " strdup()" if you need to save the
54 * filename.
55 */

```

```

114     time_t now;
115     char *buffer;
116     unsigned int size;
117
118     if (argc < 2 || get_sha1_hex(argv[1], tree_sha1) < 0)
119         usage("commit-tree <sha1> [-p <sha1>]* < changelog>");
120
121     for (i = 2; i < argc; i += 2) {
122         char *a, *b;
123         a = argv[i]; b = argv[i+1];
124         if (!b || strcmp(a, "-p") || get_sha1_hex(b, parent_sha1[parents]))
125             usage("commit-tree <sha1> [-p <sha1>]* < changelog>");
126         parents++;
127     }
128     if (!parents)
129         fprintf(stderr, "Committing initial tree %s\n", argv[1]);
130     pw = getpwuid(getuid());
131     if (!pw)
132         usage("You don't exist. Go away!");
133     realgecos = pw->pw_gecos;
134     len = strlen(pw->pw_name);
135     memcpy(realemail, pw->pw_name, len);
136     realemail[len] = '@';
137     gethostname(realemail+len+1, sizeof(realemail)-len-1);
138     time(&now);
139     realdate = ctime(&now);
140
141     gecos = getenv("COMMITTER_NAME") ? : realgecos;
142     email = getenv("COMMITTER_EMAIL") ? : realemail;
143     date = getenv("COMMITTER_DATE") ? : realdate;
144
145     remove_special(gecos); remove_special(realgecos);
146     remove_special(email); remove_special(realemail);
147     remove_special(date); remove_special(realdate);
148
149     init_buffer(&buffer, &size);
150     add_buffer(&buffer, &size, "tree %s\n", sha1_to_hex(tree_sha1));
151
152     /*
153      * NOTE! This ordering means that the same exact tree merged with a
154      * different order of parents will be a _different_ changeset even
155      * if everything else stays the same.
156      */
157     for (i = 0; i < parents; i++)
158         add_buffer(&buffer, &size, "parent %s\n", sha1_to_hex(parent_sha1[i]));
159
160     /* Person/date information */
161     add_buffer(&buffer, &size, "author %s <%s> %s\n", gecos, email, date);
162     add_buffer(&buffer, &size, "committer %s <%s> %s\n", realgecos, realemail, realdate);
163
164     /* And add the comment */
165     while (fgets(comment, sizeof(comment), stdin) != NULL)
166         add_buffer(&buffer, &size, "%s", comment);
167
168     finish_buffer("commit ", &buffer, &size);
169
170     write_sha1_file(buffer, size);
171     return 0;

```

```

56     */
57     static void finish_buffer(char *tag, char **bufp, unsigned int *sizep)
58     {
59         int i;
60         static char *name, *base;
61         int offset;
62         if (base) {
63             char *sh1_file_directory = getenv(DB_ENVIRONMENT) ?: DEFAULT_DB_ENVIRONMENT,
64             int len = strlen(sh1_file_directory);
65             base = malloc(len + 60);
66             memcpyp(base, sh1_file_directory, len);
67             memset(base + len, 0, 60);
68             base[LEN+3] = '/'; base[LEN] = '/';
69             memcpy(buf, tag, tagLen);
70             name = base + len + 1;
71             for (i = 0; i < 20; i++) {
72                 static char hex[] = "0123456789abdef";
73                 unsigned int val = sh1[i];
74                 unsigned int taglen = strlen(hex);
75                 pos = name + i*2 + (i < 0)? 0 : taglen;
76                 pos += taglen;
77                 pos = hex[val % 0x1f];
78                 pos += hex[val / 0x1f];
79             }
80         }
81         return base;
82     void *read_sh1_file(unsigned char *sh1, char *type, unsigned long *size)
83     {
84         static stream z_stream;
85         z_stream.zalloc = Z_NULL;
86         z_stream.zfree = Z_NULL;
87         z_stream.next_in = Z_NULL;
88         z_stream.avail_in = 0;
89         z_stream.next_out = Z_NULL;
90         z_stream.avail_out = type->size;
91         inflateInit(&zstream);
92         ret = inflate(&zstream, 0);
93         if (ret == Z_OK) {
94             map = mmap(NULL, st.size, PROT_READ, MAP_PRIVATE, fd, 0);
95             if (fd != open(filename, O_RDONLY));
96                 perror(filename);
97             if (fd < 0) {
98                 perror("fd");
99                 close(fd);
100            map = mmap(NULL, st.size, PROT_READ, MAP_PRIVATE, fd, 0);
101        }
102        if (-1 == (int)long(map))
103            perror("map");
104        /* Get the data stream */
105        /* Get the data stream */
106        return NULL;
107    }
108    unsigned char parent_sh1[20];
109    unsigned char parent_sha1[20];
110    char *gecos, tree_gecos;
111    char *date, realdate;
112    char comment[1000];
113    struct passwd *pw;
114
115    if (i < len) {
116        int parents = 0;
117        int main(int argc, char **argv)
118        {
119            if (parents == 0) {
120                unsigned char parent_sha1[20];
121                unsigned char parent_sha1[20];
122                char comment[1000];
123                char date, realdate;
124                char *gecos, tree_gecos;
125                char *date, realdate;
126                char *email, realemail[1000];
127                if (getenv("MAIL"))
128                    email = getenv("MAIL");
129                else
130                    email = "szjoo@butter";
131                if (getenv("HOME"))
132                    home = getenv("HOME");
133                else
134                    home = "/tmp";
135                if (getenv("LOGNAME"))
136                    logname = getenv("LOGNAME");
137                else
138                    logname = "szjoo";
139                if (getenv("USER"))
140                    user = getenv("USER");
141                else
142                    user = "szjoo";
143                if (getenv("HOSTNAME"))
144                    host = getenv("HOSTNAME");
145                else
146                    host = "szjoo";
147                if (getenv("TERM"))
148                    term = getenv("TERM");
149                else
150                    term = "xterm";
151                if (getenv("SHELL"))
152                    shell = getenv("SHELL");
153                else
154                    shell = "/bin/sh";
155                if (getenv("PWD"))
156                    pwd = getenv("PWD");
157                else
158                    pwd = "/";
159                if (getenv("HOME"))
160                    home = getenv("HOME");
161                else
162                    home = "/";
163                if (getenv("LOGNAME"))
164                    logname = getenv("LOGNAME");
165                else
166                    logname = "szjoo";
167                if (getenv("USER"))
168                    user = getenv("USER");
169                else
170                    user = "szjoo";
171                if (getenv("HOSTNAME"))
172                    host = getenv("HOSTNAME");
173                else
174                    host = "szjoo";
175                if (getenv("TERM"))
176                    term = getenv("TERM");
177                else
178                    term = "xterm";
179                if (getenv("SHELL"))
180                    shell = getenv("SHELL");
181                else
182                    shell = "/bin/sh";
183                if (getenv("PWD"))
184                    pwd = getenv("PWD");
185                else
186                    pwd = "/";
187                if (getenv("HOME"))
188                    home = getenv("HOME");
189                else
190                    home = "/";
191                if (getenv("LOGNAME"))
192                    logname = getenv("LOGNAME");
193                else
194                    logname = "szjoo";
195                if (getenv("USER"))
196                    user = getenv("USER");
197                else
198                    user = "szjoo";
199                if (getenv("HOSTNAME"))
200                    host = getenv("HOSTNAME");
201                else
202                    host = "szjoo";
203                if (getenv("TERM"))
204                    term = getenv("TERM");
205                else
206                    term = "xterm";
207                if (getenv("SHELL"))
208                    shell = getenv("SHELL");
209                else
210                    shell = "/bin/sh";
211                if (getenv("PWD"))
212                    pwd = getenv("PWD");
213                else
214                    pwd = "/";
215                if (getenv("HOME"))
216                    home = getenv("HOME");
217                else
218                    home = "/";
219                if (getenv("LOGNAME"))
220                    logname = getenv("LOGNAME");
221                else
222                    logname = "szjoo";
223                if (getenv("USER"))
224                    user = getenv("USER");
225                else
226                    user = "szjoo";
227                if (getenv("HOSTNAME"))
228                    host = getenv("HOSTNAME");
229                else
230                    host = "szjoo";
231                if (getenv("TERM"))
232                    term = getenv("TERM");
233                else
234                    term = "xterm";
235                if (getenv("SHELL"))
236                    shell = getenv("SHELL");
237                else
238                    shell = "/bin/sh";
239                if (getenv("PWD"))
240                    pwd = getenv("PWD");
241                else
242                    pwd = "/";
243                if (getenv("HOME"))
244                    home = getenv("HOME");
245                else
246                    home = "/";
247                if (getenv("LOGNAME"))
248                    logname = getenv("LOGNAME");
249                else
250                    logname = "szjoo";
251                if (getenv("USER"))
252                    user = getenv("USER");
253                else
254                    user = "szjoo";
255                if (getenv("HOSTNAME"))
256                    host = getenv("HOSTNAME");
257                else
258                    host = "szjoo";
259                if (getenv("TERM"))
260                    term = getenv("TERM");
261                else
262                    term = "xterm";
263                if (getenv("SHELL"))
264                    shell = getenv("SHELL");
265                else
266                    shell = "/bin/sh";
267                if (getenv("PWD"))
268                    pwd = getenv("PWD");
269                else
270                    pwd = "/";
271                if (getenv("HOME"))
272                    home = getenv("HOME");
273                else
274                    home = "/";
275                if (getenv("LOGNAME"))
276                    logname = getenv("LOGNAME");
277                else
278                    logname = "szjoo";
279                if (getenv("USER"))
280                    user = getenv("USER");
281                else
282                    user = "szjoo";
283                if (getenv("HOSTNAME"))
284                    host = getenv("HOSTNAME");
285                else
286                    host = "szjoo";
287                if (getenv("TERM"))
288                    term = getenv("TERM");
289                else
290                    term = "xterm";
291                if (getenv("SHELL"))
292                    shell = getenv("SHELL");
293                else
294                    shell = "/bin/sh";
295                if (getenv("PWD"))
296                    pwd = getenv("PWD");
297                else
298                    pwd = "/";
299                if (getenv("HOME"))
300                    home = getenv("HOME");
301                else
302                    home = "/";
303                if (getenv("LOGNAME"))
304                    logname = getenv("LOGNAME");
305                else
306                    logname = "szjoo";
307                if (getenv("USER"))
308                    user = getenv("USER");
309                else
310                    user = "szjoo";
311                if (getenv("HOSTNAME"))
312                    host = getenv("HOSTNAME");
313                else
314                    host = "szjoo";
315                if (getenv("TERM"))
316                    term = getenv("TERM");
317                else
318                    term = "xterm";
319                if (getenv("SHELL"))
320                    shell = getenv("SHELL");
321                else
322                    shell = "/bin/sh";
323                if (getenv("PWD"))
324                    pwd = getenv("PWD");
325                else
326                    pwd = "/";
327                if (getenv("HOME"))
328                    home = getenv("HOME");
329                else
330                    home = "/";
331                if (getenv("LOGNAME"))
332                    logname = getenv("LOGNAME");
333                else
334                    logname = "szjoo";
335                if (getenv("USER"))
336                    user = getenv("USER");
337                else
338                    user = "szjoo";
339                if (getenv("HOSTNAME"))
340                    host = getenv("HOSTNAME");
341                else
342                    host = "szjoo";
343                if (getenv("TERM"))
344                    term = getenv("TERM");
345                else
346                    term = "xterm";
347                if (getenv("SHELL"))
348                    shell = getenv("SHELL");
349                else
350                    shell = "/bin/sh";
351                if (getenv("PWD"))
352                    pwd = getenv("PWD");
353                else
354                    pwd = "/";
355                if (getenv("HOME"))
356                    home = getenv("HOME");
357                else
358                    home = "/";
359                if (getenv("LOGNAME"))
360                    logname = getenv("LOGNAME");
361                else
362                    logname = "szjoo";
363                if (getenv("USER"))
364                    user = getenv("USER");
365                else
366                    user = "szjoo";
367                if (getenv("HOSTNAME"))
368                    host = getenv("HOSTNAME");
369                else
370                    host = "szjoo";
371                if (getenv("TERM"))
372                    term = getenv("TERM");
373                else
374                    term = "xterm";
375                if (getenv("SHELL"))
376                    shell = getenv("SHELL");
377                else
378                    shell = "/bin/sh";
379                if (getenv("PWD"))
380                    pwd = getenv("PWD");
381                else
382                    pwd = "/";
383                if (getenv("HOME"))
384                    home = getenv("HOME");
385                else
386                    home = "/";
387                if (getenv("LOGNAME"))
388                    logname = getenv("LOGNAME");
389                else
390                    logname = "szjoo";
391                if (getenv("USER"))
392                    user = getenv("USER");
393                else
394                    user = "szjoo";
395                if (getenv("HOSTNAME"))
396                    host = getenv("HOSTNAME");
397                else
398                    host = "szjoo";
399                if (getenv("TERM"))
400                    term = getenv("TERM");
401                else
402                    term = "xterm";
403                if (getenv("SHELL"))
404                    shell = getenv("SHELL");
405                else
406                    shell = "/bin/sh";
407                if (getenv("PWD"))
408                    pwd = getenv("PWD");
409                else
410                    pwd = "/";
411                if (getenv("HOME"))
412                    home = getenv("HOME");
413                else
414                    home = "/";
415                if (getenv("LOGNAME"))
416                    logname = getenv("LOGNAME");
417                else
418                    logname = "szjoo";
419                if (getenv("USER"))
420                    user = getenv("USER");
421                else
422                    user = "szjoo";
423                if (getenv("HOSTNAME"))
424                    host = getenv("HOSTNAME");
425                else
426                    host = "szjoo";
427                if (getenv("TERM"))
428                    term = getenv("TERM");
429                else
430                    term = "xterm";
431                if (getenv("SHELL"))
432                    shell = getenv("SHELL");
433                else
434                    shell = "/bin/sh";
435                if (getenv("PWD"))
436                    pwd = getenv("PWD");
437                else
438                    pwd = "/";
439                if (getenv("HOME"))
440                    home = getenv("HOME");
441                else
442                    home = "/";
443                if (getenv("LOGNAME"))
444                    logname = getenv("LOGNAME");
445                else
446                    logname = "szjoo";
447                if (getenv("USER"))
448                    user = getenv("USER");
449                else
450                    user = "szjoo";
451                if (getenv("HOSTNAME"))
452                    host = getenv("HOSTNAME");
453                else
454                    host = "szjoo";
455                if (getenv("TERM"))
456                    term = getenv("TERM");
457                else
458                    term = "xterm";
459                if (getenv("SHELL"))
460                    shell = getenv("SHELL");
461                else
462                    shell = "/bin/sh";
463                if (getenv("PWD"))
464                    pwd = getenv("PWD");
465                else
466                    pwd = "/";
467                if (getenv("HOME"))
468                    home = getenv("HOME");
469                else
470                    home = "/";
471                if (getenv("LOGNAME"))
472                    logname = getenv("LOGNAME");
473                else
474                    logname = "szjoo";
475                if (getenv("USER"))
476                    user = getenv("USER");
477                else
478                    user = "szjoo";
479                if (getenv("HOSTNAME"))
480                    host = getenv("HOSTNAME");
481                else
482                    host = "szjoo";
483                if (getenv("TERM"))
484                    term = getenv("TERM");
485                else
486                    term = "xterm";
487                if (getenv("SHELL"))
488                    shell = getenv("SHELL");
489                else
490                    shell = "/bin/sh";
491                if (getenv("PWD"))
492                    pwd = getenv("PWD");
493                else
494                    pwd = "/";
495                if (getenv("HOME"))
496                    home = getenv("HOME");
497                else
498                    home = "/";
499                if (getenv("LOGNAME"))
500                    logname = getenv("LOGNAME");
501                else
502                    logname = "szjoo";
503                if (getenv("USER"))
504                    user = getenv("USER");
505                else
506                    user = "szjoo";
507                if (getenv("HOSTNAME"))
508                    host = getenv("HOSTNAME");
509                else
510                    host = "szjoo";
511                if (getenv("TERM"))
512                    term = getenv("TERM");
513                else
514                    term = "xterm";
515                if (getenv("SHELL"))
516                    shell = getenv("SHELL");
517                else
518                    shell = "/bin/sh";
519                if (getenv("PWD"))
520                    pwd = getenv("PWD");
521                else
522                    pwd = "/";
523                if (getenv("HOME"))
524                    home = getenv("HOME");
525                else
526                    home = "/";
527                if (getenv("LOGNAME"))
528                    logname = getenv("LOGNAME");
529                else
530                    logname = "szjoo";
531                if (getenv("USER"))
532                    user = getenv("USER");
533                else
534                    user = "szjoo";
535                if (getenv("HOSTNAME"))
536                    host = getenv("HOSTNAME");
537                else
538                    host = "szjoo";
539                if (getenv("TERM"))
540                    term = getenv("TERM");
541                else
542                    term = "xterm";
543                if (getenv("SHELL"))
544                    shell = getenv("SHELL");
545                else
546                    shell = "/bin/sh";
547                if (getenv("PWD"))
548                    pwd = getenv("PWD");
549                else
550                    pwd = "/";
551                if (getenv("HOME"))
552                    home = getenv("HOME");
553                else
554                    home = "/";
555                if (getenv("LOGNAME"))
556                    logname = getenv("LOGNAME");
557                else
558                    logname = "szjoo";
559                if (getenv("USER"))
560                    user = getenv("USER");
561                else
562                    user = "szjoo";
563                if (getenv("HOSTNAME"))
564                    host = getenv("HOSTNAME");
565                else
566                    host = "szjoo";
567                if (getenv("TERM"))
568                    term = getenv("TERM");
569                else
570                    term = "xterm";
571                if (getenv("SHELL"))
572                    shell = getenv("SHELL");
573                else
574                    shell = "/bin/sh";
575                if (getenv("PWD"))
576                    pwd = getenv("PWD");
577                else
578                    pwd = "/";
579                if (getenv("HOME"))
580                    home = getenv("HOME");
581                else
582                    home = "/";
583                if (getenv("LOGNAME"))
584                    logname = getenv("LOGNAME");
585                else
586                    logname = "szjoo";
587                if (getenv("USER"))
588                    user = getenv("USER");
589                else
590                    user = "szjoo";
591                if (getenv("HOSTNAME"))
592                    host = getenv("HOSTNAME");
593                else
594                    host = "szjoo";
595                if (getenv("TERM"))
596                    term = getenv("TERM");
597                else
598                    term = "xterm";
599                if (getenv("SHELL"))
600                    shell = getenv("SHELL");
601                else
602                    shell = "/bin/sh";
603                if (getenv("PWD"))
604                    pwd = getenv("PWD");
605                else
606                    pwd = "/";
607                if (getenv("HOME"))
608                    home = getenv("HOME");
609                else
610                    home = "/";
611                if (getenv("LOGNAME"))
612                    logname = getenv("LOGNAME");
613                else
614                    logname = "szjoo";
615                if (getenv("USER"))
616                    user = getenv("USER");
617                else
618                    user = "szjoo";
619                if (getenv("HOSTNAME"))
620                    host = getenv("HOSTNAME");
621                else
622                    host = "szjoo";
623                if (getenv("TERM"))
624                    term = getenv("TERM");
625                else
626                    term = "xterm";
627                if (getenv("SHELL"))
628                    shell = getenv("SHELL");
629                else
630                    shell = "/bin/sh";
631                if (getenv("PWD"))
632                    pwd = getenv("PWD");
633                else
634                    pwd = "/";
635                if (getenv("HOME"))
636                    home = getenv("HOME");
637                else
638                    home = "/";
639                if (getenv("LOGNAME"))
640                    logname = getenv("LOGNAME");
641                else
642                    logname = "szjoo";
643                if (getenv("USER"))
644                    user = getenv("USER");
645                else
646                    user = "szjoo";
647                if (getenv("HOSTNAME"))
648                    host = getenv("HOSTNAME");
649                else
650                    host = "szjoo";
651                if (getenv("TERM"))
652                    term = getenv("TERM");
653                else
654                    term = "xterm";
655                if (getenv("SHELL"))
656                    shell = getenv("SHELL");
657                else
658                    shell = "/bin/sh";
659                if (getenv("PWD"))
660                    pwd = getenv("PWD");
661                else
662                    pwd = "/";
663                if (getenv("HOME"))
664                    home = getenv("HOME");
665                else
666                    home = "/";
667                if (getenv("LOGNAME"))
668                    logname = getenv("LOGNAME");
669                else
670                    logname = "szjoo";
671                if (getenv("USER"))
672                    user = getenv("USER");
673                else
674                    user = "szjoo";
675                if (getenv("HOSTNAME"))
676                    host = getenv("HOSTNAME");
677                else
678                    host = "szjoo";
679                if (getenv("TERM"))
680                    term = getenv("TERM");
681                else
682                    term = "xterm";
683                if (getenv("SHELL"))
684                    shell = getenv("SHELL");
685                else
686                    shell = "/bin/sh";
687                if (getenv("PWD"))
688                    pwd = getenv("PWD");
689                else
690                    pwd = "/";
691                if (getenv("HOME"))
692                    home = getenv("HOME");
693                else
694                    home = "/";
695                if (getenv("LOGNAME"))
696                    logname = getenv("LOGNAME");
697                else
698                    logname = "szjoo";
699                if (getenv("USER"))
700                    user = getenv("USER");
701                else
702                    user = "szjoo";
703                if (getenv("HOSTNAME"))
704                    host = getenv("HOSTNAME");
705                else
706                    host = "szjoo";
707                if (getenv("TERM"))
708                    term = getenv("TERM");
709                else
710                    term = "xterm";
711                if (getenv("SHELL"))
712                    shell = getenv("SHELL");
713                else
714                    shell = "/bin/sh";
715                if (getenv("PWD"))
716                    pwd = getenv("PWD");
717                else
718                    pwd = "/";
719                if (getenv("HOME"))
720                    home = getenv("HOME");
721                else
722                    home = "/";
723                if (getenv("LOGNAME"))
724                    logname = getenv("LOGNAME");
725                else
726                    logname = "szjoo";
727                if (getenv("USER"))
728                    user = getenv("USER");
729                else
730                    user = "szjoo";
731                if (getenv("HOSTNAME"))
732                    host = getenv("HOSTNAME");
733                else
734                    host = "szjoo";
735                if (getenv("TERM"))
736                    term = getenv("TERM");
737                else
738                    term = "xterm";
739                if (getenv("SHELL"))
740                    shell = getenv("SHELL");
741                else
742                    shell = "/bin/sh";
743                if (getenv("PWD"))
744                    pwd = getenv("PWD");
745                else
746                    pwd = "/";
747                if (getenv("HOME"))
748                    home = getenv("HOME");
749                else
750                    home = "/";
751                if (getenv("LOGNAME"))
752                    logname = getenv("LOGNAME");
753                else
754                    logname = "szjoo";
755                if (getenv("USER"))
756                    user = getenv("USER");
757                else
758                    user = "szjoo";
759                if (getenv("HOSTNAME"))
760                    host = getenv("HOSTNAME");
761                else
762                    host = "szjoo";
763                if (getenv("TERM"))
764                    term = getenv("TERM");
765                else
766                    term = "xterm";
767                if (getenv("SHELL"))
768                    shell = getenv("SHELL");
769                else
770                    shell = "/bin/sh";
771                if (getenv("PWD"))
772                    pwd = getenv("PWD");
773                else
774                    pwd = "/";
775                if (getenv("HOME"))
776                    home = getenv("HOME");
777                else
778                    home = "/";
779                if (getenv("LOGNAME"))
780                    logname = getenv("LOGNAME");
781                else
782                    logname = "szjoo";
783                if (getenv("USER"))
784                    user = getenv("USER");
785                else
786                    user = "szjoo";
787                if (getenv("HOSTNAME"))
788                    host = getenv("HOSTNAME");
789                else
790                    host = "szjoo";
791                if (getenv("TERM"))
792                    term = getenv("TERM");
793                else
794                    term = "xterm";
795                if (getenv("SHELL"))
796                    shell = getenv("SHELL");
797                else
798                    shell = "/bin/sh";
799                if (getenv("PWD"))
800                    pwd = getenv("PWD");
801                else
802                    pwd = "/";
803                if (getenv("HOME"))
804                    home = getenv("HOME");
805                else
806                    home = "/";
807                if (getenv("LOGNAME"))
808                    logname = getenv("LOGNAME");
809                else
810                    logname = "szjoo";
811                if (getenv("USER"))
812                    user = getenv("USER");
813                else
814                    user = "szjoo";
815                if (getenv("HOSTNAME"))
816                    host = getenv("HOSTNAME");
817                else
818                    host = "szjoo";
819                if (getenv("TERM"))
820                    term = getenv("TERM");
821                else
822                    term = "xterm";
823                if (getenv("SHELL"))
824                    shell = getenv("SHELL");
825                else
826                    shell = "/bin/sh";
827                if (getenv("PWD"))
828                    pwd = getenv("PWD");
829                else
830                    pwd = "/";
831                if (getenv("HOME"))
832                    home = getenv("HOME");
833                else
834                    home = "/";
835                if (getenv("LOGNAME"))
836                    logname = getenv("LOGNAME");
837                else
838                    logname = "szjoo";
839                if (getenv("USER"))
840                    user = getenv("USER");
841                else
842                    user = "szjoo";
843                if (getenv("HOSTNAME"))
844                    host = getenv("HOSTNAME");
845                else
846                    host = "szjoo";
847                if (getenv("TERM"))
848                    term = getenv("TERM");
849                else
850                    term = "xterm";
851                if (getenv("SHELL"))
852                    shell = getenv("SHELL");
853                else
854                    shell = "/bin/sh";
855                if (getenv("PWD"))
856                    pwd = getenv("PWD");
857                else
858                    pwd = "/";
859                if (getenv("HOME"))
860                    home = getenv("HOME");
861                else
862                    home = "/";
863                if (getenv("LOGNAME"))
864                    logname = getenv("LOGNAME");
865                else
866                    logname = "szjoo";
867                if (getenv("USER"))
868                    user = getenv("USER");
869                else
870                    user = "szjoo";
871                if (getenv("HOSTNAME"))
872                    host = getenv("HOSTNAME");
873                else
874                    host = "szjoo";
875                if (getenv("TERM"))
876                    term = getenv("TERM");
877                else
878                    term = "xterm";
879                if (getenv("SHELL"))
880                    shell = getenv("SHELL");
881                else
882                    shell = "/bin/sh";
883                if (getenv("PWD"))
884                    pwd = getenv("PWD");
885                else
886                    pwd = "/";
887                if (getenv("HOME"))
888                    home = getenv("HOME");
889                else
890                    home = "/";
891                if (getenv("LOGNAME"))
892                    logname = getenv("LOGNAME");
893                else
894                    logname = "szjoo";
895                if (getenv("USER"))
896                    user = getenv("USER");
897                else
898                    user = "szjoo";
899                if (getenv("HOSTNAME"))
900                    host = getenv("HOSTNAME");
901                else
902                    host = "szjoo";
903                if (getenv("TERM"))
904                    term = getenv("TERM");
905                else
906                    term = "xterm";
907                if (getenv("SHELL"))
908                    shell = getenv("SHELL");
909                else
910                    shell = "/bin/sh";
911                if (getenv("PWD"))
912                    pwd = getenv("PWD");
913                else
914                    pwd = "/";
915                if (getenv("HOME"))
916                    home = getenv("HOME");
917                else
918                    home = "/";
919                if (getenv("LOGNAME"))
920                    logname = getenv("LOGNAME");
921                else
922                    logname = "szjoo";
923                if (getenv("USER"))
924                    user = getenv("USER");
925                else
926                    user = "szjoo";
927                if (getenv("HOSTNAME"))
928                    host = getenv("HOSTNAME");
929                else
930                    host = "szjoo";
931                if (getenv("TERM"))
932                    term = getenv("TERM");
933                else
934                    term = "xterm";
935                if (getenv("SHELL"))
936                    shell = getenv("SHELL");
937                else
938                    shell = "/bin/sh";
939                if (getenv("PWD"))
940                    pwd = getenv("PWD");
941                else
942                    pwd = "/";
943                if (getenv("HOME"))
944                    home = getenv("HOME");
945                else
946                    home = "/";
947                if (getenv("LOGNAME"))
948                    logname = getenv("LOGNAME");
949                else
950                    logname = "szjoo";
951                if (getenv("USER"))
952                    user = getenv("USER");
953                else
954                    user = "szjoo";
955                if (getenv("HOSTNAME"))
956                    host = getenv("HOSTNAME");
957                else
958                    host = "szjoo";
959                if (getenv("TERM"))
960                    term = getenv("TERM");
961                else
962                    term = "xterm";
963                if (getenv("SHELL"))
964                    shell = getenv("SHELL");
965                else
966                    shell = "/bin/sh";
967                if (getenv("PWD"))
968                    pwd = getenv("PWD");
969                else
970                    pwd = "/";
971                if (getenv("HOME"))
972                    home = getenv("HOME");
973                else
974                    home = "/";
975                if (getenv("LOGNAME"))
976                    logname = getenv("LOGNAME");
977                else
978                    logname = "szjoo";
979                if (getenv("USER"))
980                    user = getenv("USER");
981                else
982                    user = "szjoo";
983                if (getenv("HOSTNAME"))
984                    host = getenv("HOSTNAME");
985                else
986                    host = "szjoo";
987                if (getenv("TERM"))
988                    term = getenv("TERM");
989                else
990                    term = "xterm";
991                if (getenv("SHELL"))
992                    shell = getenv("SHELL");
993                else
994                    shell = "/bin/sh";
995                if (getenv("PWD"))
996                    pwd = getenv("PWD");
997                else
998                    pwd = "/";
999                if (getenv("HOME"))
1000                    home = getenv("HOME");
1001                else
1002                    home = "/";
1003                if (getenv("LOGNAME"))
1004                    logname = getenv("LOGNAME");
1005                else
1006                    logname = "szjoo";
1007                if (getenv("USER"))
1008                    user = getenv("USER");
1009                else
1010                    user = "szjoo";
1011                if (getenv("HOSTNAME"))
1012                    host = getenv("HOSTNAME");
1013                else
1014                    host = "szjoo";
1015                if (getenv("TERM"))
1016                    term = getenv("TERM");
1017                else
1018                    term = "xterm";
1019                if (getenv("SHELL"))
1020                    shell = getenv("SHELL");
1021                else
1022                    shell = "/bin/sh";
1023                if (getenv("PWD"))
1024                    pwd = getenv("PWD");
1025                else
1026                    pwd = "/";
1027                if (getenv("HOME"))
1028                    home = getenv("HOME");
1029                else
1030                    home = "/";
1031                if (getenv("LOGNAME"))
1032                    logname = getenv("LOGNAME");
1033                else
1034                    logname = "szjoo";
1035                if (getenv("USER"))
1036                    user = getenv("USER");
1037                else
1038                    user = "szjoo";
1039                if (getenv("HOSTNAME"))
1040                    host = getenv("HOSTNAME");
1041                else
1042                    host = "szjoo";
1043                if (getenv("TERM"))
1044                    term = getenv("TERM");
1045                else
1046                    term = "xterm";
1047                if (getenv("SHELL"))
1048                    shell = getenv("SHELL");
1049                else
1050                    shell = "/bin/sh";
1051                if (getenv("PWD"))
1052                    pwd = getenv("PWD");
1053                else
1054                    pwd = "/";
1055                if (getenv("HOME"))
1056                    home = getenv("HOME");
1057                else
1058                    home = "/";
1059                if (getenv("LOGNAME"))
1060                    logname = getenv("LOGNAME");
1061                else
1062                    logname = "szjoo";
1063                if (getenv("USER"))
1064                    user = getenv("USER");
1065                else
1066                    user = "szjoo";
1067                if (getenv("HOSTNAME"))
1068                    host = getenv("HOSTNAME");
1069                else
1070                    host = "szjoo";
1071                if (getenv("TERM"))
1072                    term = getenv("TERM");
1073                else
1074                    term = "xterm";
1075                if (getenv("SHELL"))
1076                    shell = getenv("SHELL");
1077                else
1078                    shell = "/bin/sh";
1079                if (getenv("PWD"))
1080                    pwd = getenv("PWD");
1081                else
1082                    pwd = "/";
1083                if (getenv("HOME"))
1084                    home = getenv("HOME");
1085                else
1086                    home = "/";
1087                if (getenv("LOGNAME"))
1088                    logname = getenv("LOGNAME");
1089                else
1090                    logname = "szjoo";
1091                if (getenv("USER"))
1092                    user = getenv("USER");
1093                else
1094                    user = "szjoo";
1095                if (getenv("HOSTNAME"))
1096                    host = getenv("HOSTNAME");
1097                else
1098                    host = "szjoo";
1099                if (getenv("TERM"))
1100                    term = getenv("TERM");
1101                else
1102                    term = "xterm";
1103                if (getenv("SHELL"))
1104                    shell = getenv("SHELL");
1105                else
1106                    shell = "/bin/sh";
1107                if (getenv("PWD"))
1108                    pwd = getenv("PWD");
1109                else
1110                    pwd = "/";
1111                if (getenv("HOME"))
1112                    home = getenv("HOME");
1113                else
1114                    home = "/";
1115                if (getenv("LOGNAME"))
1116                    logname = getenv("LOGNAME");
1117                else
1118                    logname = "szjoo";
1119                if (getenv("USER"))
1120                    user = getenv("USER");
1121                else
1122                    user = "szjoo";
1123                if (getenv("HOSTNAME"))
1124                    host = getenv("HOSTNAME");
1125                else
1126                    host = "szjoo";
1127                if (getenv("TERM"))
1128                    term = getenv("TERM");
1129                else
1130                    term = "xterm";
1131                if (getenv("SHELL"))
1132                    shell = getenv("SHELL");
1133                else
1134                    shell = "/bin/sh";
1135                if (getenv("PWD"))
1136                    pwd = getenv("PWD");
1137                else
1138                    pwd = "/";
1139                if (getenv("HOME"))
1140                    home = getenv("HOME");
1141                else
1142                    home = "/";
1143                if (getenv("LOGNAME"))
1144                    logname = getenv("LOGNAME");
1145                else
1146                    logname = "szjoo";
1147                if (getenv("USER"))
1148                    user = getenv("USER");
1149                else
1150                    user = "szjoo";
1151                if (getenv("HOSTNAME"))
1152                    host = getenv("HOSTNAME");
1153                else
1154                    host = "szjoo";
1155                if (getenv("TERM"))
1156                    term = getenv("TERM");
1157                else
1158                    term = "xterm";
1159                if (getenv("SHELL"))
1160                    shell = getenv("SHELL");
1161                else
1162                    shell = "/bin/sh";
1163                if (getenv("PWD"))
1164                    pwd = getenv("PWD");
1165                else
1166                    pwd = "/";
1167                if (getenv("HOME"))
1168                    home = getenv("HOME");
1169                else
1170                    home = "/";
1171                if (getenv("LOGNAME"))
1172                    logname = getenv("LOGNAME");
1173                else
1174                    logname = "szjoo";
1175                if (getenv("USER"))
1176                    user = getenv("USER");
1177                else
1178                    user = "szjoo";
1179                if (getenv("HOSTNAME"))
1180                    host = getenv("HOSTNAME");
1181                else
1182                    host = "szjoo";
1183                if (getenv("TERM"))
1184                    term = getenv("TERM");
1185                else
1186                    term = "xterm";
1187                if (getenv("SHELL"))
1188                    shell = getenv("SHELL");
1189                else
1190                    shell = "/bin/sh";
1191                if (getenv("PWD"))
1192                    pwd = getenv("PWD");
1193                else
1194                    pwd = "/";

```

```

114 if (sscanf(buffer, "%10s %lu", type, size) != 2)
115     return NULL;
116 bytes = strlen(buffer) + 1;
117 buf = malloc(*size);
118 if (!buf)
119     return NULL;
120
121 memcpy(buf, buffer + bytes, stream.total_out - bytes);
122 bytes = stream.total_out - bytes;
123 if (bytes < *size && ret == Z_OK) {
124     stream.next_out = buf + bytes;
125     stream.avail_out = *size - bytes;
126     while (inflate(&stream, Z_FINISH) == Z_OK)
127         /* nothing */;
128 }
129 inflateEnd(&stream);
130 return buf;
131 }

132 int write_sha1_file(char *buf, unsigned len)
133 {
134     int size;
135     char *compressed;
136     z_stream stream;
137     unsigned char sha1[20];
138     SHA_CTX c;
139
140     /* Set it up */
141     memset(&stream, 0, sizeof(stream));
142     deflateInit(&stream, Z_BEST_COMPRESSION);
143     size = deflateBound(&stream, len);
144     compressed = malloc(size);
145
146     /* Compress it */
147     stream.next_in = buf;
148     stream.avail_in = len;
149     stream.next_out = compressed;
150     stream.avail_out = size;
151     while (deflate(&stream, Z_FINISH) == Z_OK)
152         /* nothing */;
153     deflateEnd(&stream);
154     size = stream.total_out;
155
156     /* Sha1.. */
157     SHA1_Init(&c);
158     SHA1_Update(&c, compressed, size);
159     SHA1_Final(sha1, &c);
160
161     if (write_sha1_buffer(sha1, compressed, size) < 0)
162         return -1;
163     printf("%s\n", sha1_to_hex(sha1));
164     return 0;
165 }

166 int write_sha1_buffer(unsigned char *sha1, void *buf, unsigned int size)
167 {
168     char *filename = sha1_file_name(sha1);
169     int i, fd;

```

7 commit-tree.c

```

1 #include "cache.h"
2
3 #include <pwd.h>
4 #include <time.h>
5
6 #define BLOCKING (1ul << 14)
7 #define ORIG_OFFSET (40)
8
9 /*
10  * Leave space at the beginning to insert the tag
11  * once we know how big things are.
12  *
13  * FIXME! Share the code with "write-tree.c"
14  */
15 static void init_buffer(char **bufp, unsigned int *sizep)
16 {
17     char *buf = malloc(BLOCKING);
18     memset(buf, 0, ORIG_OFFSET);
19     *sizep = ORIG_OFFSET;
20     *bufp = buf;
21 }

22
23 static void add_buffer(char **bufp, unsigned int *sizep, const char *fmt, ...)
24 {
25     char one_line[2048];
26     va_list args;
27     int len;
28     unsigned long alloc, size, newsize;
29     char *buf;

30     va_start(args, fmt);
31     len = vsnprintf(one_line, sizeof(one_line), fmt, args);
32     va_end(args);
33     size = *sizep;
34     newsize = size + len;
35     alloc = (size + 32767) & ~32767;
36     buf = *bufp;
37     if (newsize > alloc) {
38         alloc = (newsize + 32767) & ~32767;
39         buf = realloc(buf, alloc);
40         *bufp = buf;
41     }
42     *sizep = newsize;
43     memcpy(buf + size, one_line, len);
44 }

45

46 static int prepend_integer(char *buffer, unsigned val, int i)
47 {
48     buffer[--i] = '\0';
49     do {
50         buffer[-i] = '0' + (val % 10);
51         val /= 10;
52     } while (val);
53     return i;
54 }

55

```



```

230     errno = EINVAL;
231     if (size > sizeof(struct cache_header))
232         map = mmap(NULL, size, PROT_READ, MAP_PRIVATE, fd, 0);
233 }
234 close(fd);
235 if (-1 == (int)(long)map)
236     return error("mmap failed");
237
238 hdr = map;
239 if (verify_hdr(hdr, size) < 0)
240     goto unmap;
241
242 active_nr = hdr->entries;
243 active_alloc = alloc_nr(active_nr);
244 active_cache = calloc(active_alloc, sizeof(struct cache_entry *));
245
246 offset = sizeof(*hdr);
247 for (i = 0; i < hdr->entries; i++) {
248     struct cache_entry *ce = map + offset;
249     offset = offset + ce_size(ce);
250     active_cache[i] = ce;
251 }
252 return active_nr;
253
254 unmap:
255     munmap(map, size);
256     errno = EINVAL;
257     return error("verify header failed");
258 }
```

```

230     perror("unable to create new cachefile");
231     return -1;
232 }
233 for (i = 1 ; i < argc; i++) {
234     char *path = argv[i];
235     if (!verify_path(path)) {
236         fprintf(stderr, "Ignoring path %s\n", argv[i]);
237         continue;
238     }
239     if (add_file_to_cache(path)) {
240         fprintf(stderr, "Unable to add %s to database\n", path);
241         goto out;
242     }
243     if (!write_cache(newfd, active_cache, active_nr) && !rename(".dircache/index.lock", ".dircache/i
244         return 0;
245     out:
246     unlink(".dircache/index.lock");
247 }
```

update-cache.c

```

56 /* existing match? Just replace it */
57 if (pos < 0) {
58     active_cache[-pos-1] = ce;
59     return 0;
60 }
61
62 /* Make sure the array is big enough .. */
63 if (active_nr == active_alloc) {
64     active_alloc = alloc_nr(active_alloc);
65     active_cache = realloc(active_cache, active_alloc * sizeof(struct cache_entry *));
66 }
67
68 /* Add it in.. */
69 active_nr++;
70 if (active_nr > pos)
71     memmove(active_cache + pos + 1, active_cache + pos, (active_nr - pos - 1) * sizeof(ce));
72 active_cache[pos] = ce;
73 return 0;
74 }
75
76 static int index_fd(const char *path, int namelen, struct cache_entry *ce, int fd, struct stat *st)
77 {
78     z_stream stream;
79     int max_out_bytes = namelen + st->st_size + 200;
80     void *out = malloc(max_out_bytes);
81     void *metadata = malloc(namelen + 200);
82     void *in = mmap(NULL, st->st_size, PROT_READ, MAP_PRIVATE, fd, 0);
83     SHA_CTX c;
84
85     close(fd);
86     if (!out || (int)(long)in == -1)
87         return -1;
88
89     memset(&stream, 0, sizeof(stream));
90     deflateInit(&stream, Z_BEST_COMPRESSION);
91
92     /*
93      * ASCII size + nul byte
94      */
95     stream.next_in = metadata;
96     stream.avail_in = 1+sprintf(metadata, "blob %lu", (unsigned long) st->st_size);
97     stream.next_out = out;
98     stream.avail_out = max_out_bytes;
99     while (deflate(&stream, 0) == Z_OK)
100        /* nothing */;
101
102    /*
103     * File content
104     */
105    stream.next_in = in;
106    stream.avail_in = st->st_size;
107    while (deflate(&stream, Z_FINISH) == Z_OK)
108        /*nothing */;
109
110    deflateEnd(&stream);
111
112    SHA1_Init(&c);
113    SHA1_Update(&c, out, stream.total_out);

```

```

114     SHA1_Final(ce->sha1, &c);
115
116     return write_sha1_buffer(ce->sha1, out, stream.total_out);
117 }
118
119 static int add_file_to_cache(char *path)
120 {
121     int size, namelen;
122     struct cache_entry *ce;
123     struct stat st;
124     int fd;
125
126     fd = open(path, O_RDONLY);
127     if (fd < 0) {
128         if (errno == ENOENT)
129             return remove_file_from_cache(path);
130         return -1;
131     }
132     if (fstat(fd, &st) < 0) {
133         close(fd);
134         return -1;
135     }
136     namelen = strlen(path);
137     size = cache_entry_size(namelen);
138     ce = malloc(size);
139     memset(ce, 0, size);
140     memcpy(ce->name, path, namelen);
141     ce->ctime.sec = st.st_ctime;
142     ce->ctime.nsec = st.st_ctim.tv_nsec;
143     ce->mtime.sec = st.st_mtime;
144     ce->mtime.nsec = st.st_mtim.tv_nsec;
145     ce->st_dev = st.st_dev;
146     ce->st_ino = st.st_ino;
147     ce->st_mode = st.st_mode;
148     ce->st_uid = st.st_uid;
149     ce->st_gid = st.st_gid;
150     ce->st_size = st.st_size;
151     ce->namelen = namelen;
152
153     if (index_fd(path, namelen, ce, fd, &st) < 0)
154         return -1;
155
156     return add_cache_entry(ce);
157 }
158
159 static int write_cache(int newfd, struct cache_entry **cache, int entries)
160 {
161     SHA_CTX c;
162     struct cache_header hdr;
163     int i;
164
165     hdr.signature = CACHE_SIGNATURE;
166     hdr.version = 1;
167     hdr.entries = entries;
168
169     SHA1_Init(&c);
170     SHA1_Update(&c, &hdr, offsetof(struct cache_header, sha1));
171     for (i = 0; i < entries; i++) {

```