Sample project highlights (Rightcore Network Services)

Described here are projects that I have worked on as part of software business exploration (Rightcore Network Services). These projects mostly are related to information technology related: functions, software components, and utilities (for local system use or as network servers). In some cases below either or both of software utilities or software component names are provided. These names can be used if it is desired to actually peruse the associated source language program code. The code is accessible on the GitHub repository located at the URL below. Utility programs are usually, but not necessarily always, located in their own subdirectory in under the main GitHub repository. Software components are often, but not always, located in library subdirectories named either LIBDAM, LIBPCS, LIBCSDB, or under the main utility which uses the components.

        http://GitHub.com/DavidMorano/RightcoreMainBase

Additional code is located in the GitHub repository:

        http://GitHub.com/DavidMorano/misc

This additional repository can be used to view code not visible in the above repository, due to some restrictions in how much code can be viewed using the web interface of GitHub.

Approximately 1.2 million non-commented source code lines (NCSL) are available for perusal in this Git repository. Enjoy as may be interested.

--------------------- Projects ---------------------

== Network de-multiplexing server family

This project consists of a family of server programs that feature a de-multiplexing front-end and which dispatches a resulting named service using a service-dispatch database. These servers act as super-servers for sub-servers selected based on an acquired service name. These servers can be configured to listen on a variety of network sources such as:

+ UNIX domain message-access socket
+ UNIX domain sockets
+ UNIX named pipes (accepting connections)
+ UNIX named pipes (being passed connections)
+ TCP stream sockets

The latter four of these above can be repeated as necessary for additional listen access points. UNIX domain sockets and UNIX names pipes are specified by naming their corresponding files in the file system. TCP network listeners can be specified by a tuple consisting of:

+ Internet protocol (either IPv4 or IPv6)
+ an Internet host name (include ANYHOST)
+ a TCP port

Services names are acquired based on server types. Current server types, each handled by its own server program (each a variant in the family), de-multiplex incoming connections using the following listen service-acquisition protocols:

+ TCPMUX service name
+ login-name
+ login-name along with a TELNET daemon service
+ finger-name

Access control is also implemented based on the incoming connection IP address. Both a custom access database can be used along with any NIS NETGROUP data which may be present. Incoming connections which fail authentication are denied.

Dispatched sub-servers can be of several varieties:

+ an already running program which accepts a passed file descriptor
+ a dynamically loaded shared object command, otherwise looking like an
  independent program
+ an independent program (typical)
+ a special built-in service

Further, services for the server can be configured to optionally use more than one of the above sub-servers. If more than a single sub-server is configured, the first one available following a certain selection order if chosen. The selection order is in the descending order as the varieties are listed above.

== Multi-threaded integrated network de-multiplexing server

This is another super-server which provides a de-multiplexing function, similar to the previous de-multiplexing server above. But rather than internally forking itself in order to handle each possible incoming service connection, rather it simply spawns an additional internal thread. This approach provides some performance advantages over a more traditional forking server. The server provides a small number of service acquisition types (how a service name is acquired from the network client). These types are currently TCPMUX and FINGER. The server adopts a service acquisition type based on how it is initially launched. The server can be configured to listen on a variety of network sources such as:

+ UNIX domain message-access socket
+ UNIX domain sockets
+ UNIX named pipes (accepting connections)
+ UNIX named pipes (being passed connections)
+ TCP stream sockets

The latter four of these above can be repeated as necessary for additional listen access points. UNIX domain sockets and UNIX names pipes are specified by naming their corresponding files in the file system. TCP network listeners can be specified by a tuple consisting of:

+ Internet protocol (either IPv4 or IPv6)
+ an Internet host name (include ANYHOST)
+ a TCP port

Services can be configured to be dispatched using a set of varieties as follows:

+ an already running sub-server program which accepts a passed file descriptor
+ by the specification of a special file that represents a file-descriptor
  access point to a sub-server; the sub-server can be either already running
  or launched on start by access to a special middle-ware file thar serves
  to activate a sub-server on access
+ an independent program (typical)
+ a special built-in service

More than a single way to dispatch a sub-server is allowed. The super-server follows the list of available dispatch mechanisms according to the descending order in the list above, and the first of these dispatch mechanisms to be available is used to dispatch the service to a sub-server.

== File backup utilities

This project consists of file backup utilities. These utilities consist of two main facilities. The first of these performs a traditional set of backup functions. These functions consist of three elements:

+ performing full backups
+ performing incremental backups
+ deletion of old backups (both full and incremental)

These functions can be scheduled as desired by the administrator. Both full and incremental backups are compressed on the target storage (generally half-line disk storage). An example of a traditional backup schedule might consist of performing incremental backups every hour and performing full backups every day or every week. Deletion of backups are often sometime between two or three weeks, or longer depending on retention requirements. Further, the files that are selected for either incremental of full backups can be configured according to some criteria. By default, all files are selected for backup. But criteria can be specified to select files according to file-extension, file type, or by some limited file-content. Files can be either selected or rejected based on the specified criteria. This selection option allows for limiting the class of files backed up to only those important for any given organization.

The second backup facility consists of several utilities which perform a combination of source file selection and certain desired functions. There are two main utilities in this group that either perform a file-copy mirroring function or a file-link mirroring function. Files in a source file-system are mirrored to a destination file-system by either copying of the files or by linking the files. In both types, Exact duplicated file-systems are created. The file-copy version is straight forward (the source files are copied to the destination. But the file-link function is more novel in that the files in the source file-system are linked using UNIX hard links to the destination area (located on the same file-system as the source). While the copying (mirroring) of files provides traditional protection against disk drive failure, the idea of file linking provides protection against accidental file deletion by users. In many environments, file deletion can be a greater risk for data retention than disk failure. Having whole file hierarchies mirror-linked provides protection against this kind of incident. Either of these utilities (mirror-copy or mirror-link) also allow for file selection based on file-extension, file-type, and some limited file-content criteria. The utilities in this group (including the two above) are:

+ filesyncer	mirror selected files in a hierarchy (copying)
+ filelinker	link selected files in a hierarchy (linking)
+ filefind	find selected files in a hierarchy
+ filerm	remove (delete) selected files in a hierarchy
+ filesize	calculate accumulated used file space for selected files in a hierarchy

== UNIX user account management (components and utilities)

A variety of UNIX account management software components and utilities have been created. Some of these utilities consist of:

+ user information retrieval and display
+ group information retrieval and display
+ project information retrieval and display (for those systems which
  support UNIX projects)

Some of the software components are:

+ user account information retrieval emulated system call
  with per-process caching
+ multithread-safe user account information retrieval
  (w/ enumeration and system-wide caching)
+ multithread-safe group account information retrieval
  (w/ enumeration and system-wide caching)
+ multithread-safe project account information retrieval
  (w/ enumeration and system-wide caching)
+ best username determination and retrieval
  (and optionally whole user account record retrieval)

+ reverse real-name (including partial and abbreviated name) lookup facility
  consisting of various cooperating components which provide system-wide caching
  of read-name to username translations

== Name service facility

This is a project which functions to provide real names, organization names, and sub-organization names (or project names) for software program clients where only some type of user identifier is initially available. It consists of a number of software components that work together, along with various utilities, and an optional server program, to create lists of real names from three possible types of identifiers. The three identifier input types available are:

+ username (login-identifier)
+ a partial or otherwise abbreviated real name
  (possibly expanded to multiple names)
+ a group identifier (usually expanded to multiple names)

The software components include:

+ name server front-end object which provides the main interface to programs
  or utilities
+ back-end loadable module supporting the front-end object
+ a back-end name-entry caching object supporting the above objects
+ an optional server daemon which will cache name requests across
  the whole of the system (without the daemon running requests are still
  cached locally for one or more integrated front-end client objects)

Utility programs associated with this project include:

+ setting of real-name (two types supported: abbreviated and full), organization
name, and project name
+ retrieval of real-names (two types), organization name, and project name

== Enhanced UNIX login facility and management

This project enhances the normal UNIX login facility by providing system configured login environments for users who log into the system using normal login (itself depending on PAM) facilities. These include at least all terminal and network logins. The end result is a preconfigured environment for the login process consisting of pre-set environment variables. This sort of facility obviates the need to have users configure more traditional configuration mechanisms such as login-profile or login-environment (details of which depend on what is supported by any particular user shell program in use).

The environment configured by the site administrator is setup by means of a set of optional system configuration files. The first of these files specifies a number of items optionally used by the facility. These are:

+ a list of programs to run as the user login shell
+ a default login shell if all else fails
+ a list of users for whom enhanced environment management should be disabled
+ a default organization name when no other is found

The second configuration file provides the main means to augment login-process environment. This file consists of a list (specified in a certain language format) where elements can be conditionally added to environment variables in certain ways. The specification language capability is quite flexible and provides mechanisms (for example) to add elements to regular string variables, directory path-like variables, string list-like variables, and more. The mechanisms provided are tailored towards what is usually needed for login environment variable management or manipulation.

The third configuration file list those path-like environment variables which can optionally have additional processing done on them. This additional processing usually includes removing both duplicated identical string components, and the removal of duplicated directory components without regard for the specific directory name but rather using device and inode identifiers. By eliminating duplicates (of whatever sort), run-time search of these components is reduced.

== general UNIX system utilities

This project consists of both software components and utilities to facilitate system management.

Some of the software components include:

+ cached retrieval of system vendor information (in two groups, ten items
  total)
+ cached retrieval of various system OS parameters or administratively
  configured parameters
+ a set of cooperating polling components to support periodic system maintenance
  functions (pcspoll)
+ cooperating software components to provide system-wide configuration
information for programs or utilities which are a part of coordinated software
distributions or facilities (pcsconf for Persona-Communication-Services)

Some of these utilities are:

+ SYSVAL
This utility is used to retrieve and display almost any OS-defined or administrator configured element of the UNIX system. These consist of many of the "sysinfo(3c)|. |sysconf(3c)|, and |confstr(3c)| values, as well as much other miscellaneous configured parameters of the system. Also, filesystem information can be retrieved, as well as dynamic state of the system (such as load averages). See the |sysval(1)| utility for more information.

+ SYSVAR
This utility is used to provide a way, both programmatically and for the administrator, to configure and retrieve software distribution installation root directories, or other configuration strings usually associated with installed software distributions or facilities. The information is cached (system-wide) for quick access by other software components, utilities, or programs.

+ MKPWI
This is both an An administrative utility and a system daemon to create or maintenance the real-name to username translation caching database.

+ PCSPOLL or POLL
This is an administrative utility and a system daemon for use in performing periodic per-user or system-wide maintenance functions. This is very much like a conventional server daemon except that it is either invoked on the occurrence of certain conditions, or can be configured to be invoked periodically. This server also supports[two types of access control lists (NIS NETGROUP and an additional custom mechanism).

+ pcspoll software object components and loadable modules
This is a polling facility consisting of multi-threaded programmable software components (including dynamically loadable task modules) which are invoked on a periodic basis triggered by user or system events. These task modules can be used to perform periodic maintenance on the whole system without the need for standing server daemon programs. This kind of device reduces the need to have tens or hundreds of server programs or daemon running which perform primarily periodic system service or system maintenance functions.

+ SYSFS
This is a system daemon which maintenances the system middle-ware cache and index files.

+ SYSDB
This is an administrative utility to query various UNIX system databases.

== Enhanced Secure Login Server

This project starts with the open-source Secure Login server and adds two main features to the server side of the facility.

The first added feature is the use of more than one possible host or user cryptographic keys to validate an incoming secure connection. This is accomplished by allowing for more than a single key to be associated with either a user or a host. Multiple keys for users or hosts can be stored on the server in order to be used for authentication. If one key fails authentication, the server searches for a possible second key that might succeed. Only if no valid key is found that satisfies an authentication is a secure connection refused. This feature allows for users (or hosts) to create new keys while being able to continue to make secure connections with remote hosts until all old keys can be removed.

The second major enhancement to the server is the allowance for additional process environment to be provided for spawned programs or logins. This is accomplished with the introduction of several new configuration files that provide additional environment at the server side. One of these configuration files provides general (any name with any value) environment variables to be added to a spawned process. Additional configurations files provide for enhanced environment for a set number of environment variables. Some of these variables which can be enhanced include:

- PATH
- LD_LIBRARY_PATH
- MANPATH

Enhanced environment provided by the server creates a more normal environment for either spawned programs or logins. Without this enhanced environment, spawned programs have extremely minimal environmental context when they begin execution.

== Email related components and facilities

This project includes a large number of software class object components for use in email construction and interpretation, as well full email utility programs and servers. Some of these software object components include:

+ mail message environment detection and parsing
+ mail message header detection and parsing
+ mail message basic framework parsing
+ mail message file handling (of various interface varieties)
+ mail message envelope parsing
+ mail message header management
+ mail message address-header parsing
+ mail message header value extraction (various types)
+ mail message header folding (of various types)
+ mail message header decoding
+ BASE64 encode and decode
+ MIME Q decoding
+ UTF-8 decoding
+ generic character set translation
+ mail message attachment management
+ mail message minimal parsing and staging for further processing
+ mail address parsing
+ mailbox framework parsing and use management
+ mailbox message caching
+ other mailbox related components (like adding a message to a mailbox)

Full mail related utilities include (each too detailed to elaborate here):

+ COMSAT - enhanced mail notification server (local mail notification server, multithreaded)
+ DMAIL - mail delivery agent, for UNIX mail spool delivery
+ DMAILBOX - mail delivery agent, for user mailbox delivery
+ EMA - mail message header value extraction, and special additional processing for header address field extraction
+ KMMSG - non-interactive mail message construction (including attachments)
+ IMAIL - mail message insertion agent (insert new mail messages into the network)
+ PCSGETMAIL - mail message spool retrieval agent (for use by user mail-reader agents)
+ ISMAILADDR - mail utility used for local address detection (used in the pipeline for incoming mail delivery to spool area for bypassing junk mail detection)
+ MBPROC - administrative mailbox manipulation processing
+ MBEXPIRE - low-level mailbox mail-message expiration processing
+ MAILEXPIRE - system-side administrative mailbox mail-message expiration processing
+ VMAIL - a full interactive user mail agent
 (I use this for most of my own mail everyday)
+ NEWMAIL - new mail summary
+ RMAILER - mail transport injection (client) agent (special protocol)
+ RMAILERD - mail transport server (special protocol)

== UNIX adaptation layer software

This project provides a software adaptation layer for all (or almost all) of the standard UNIX section-2 kernel and section-3 (all standard varieties) system kernel calls and library calls. This code makes calling into either the UNIX kernel or the system libraries much more uniform and less error prone than the standard library interfaces. In particular, almost all of the subroutines in this software layer provide a uniform way to return thread-safe error-number (errno) information. In addition many underlying UNIX calls that can be interrupted by UNIX signals are modified to be restarted on signal interruption. This is especially important on systems where much or almost all of the POSIX standard thread library calls get aborted on reception of a UNIX signal.

This project is split into two parts. The first is to provide an adaptation layer for all of the UNIX section-2 kernel calls. The second part provides an adaptation layer for all standard UNIX system libraries. These consist of (at least) the libraries of: libc, libdl, libnsl, libsocket, libxnet, libsec, libsecdb, libproject, libpthread, librt, and others as may be present.
In all, more than about 725 kernel and system calls are abstracted into a more uniform and signal safe API. They are coded almost entirely in C language.

These adaptation layers form the basis (albeit a low-level very humble one) for much of the business oriented production that has been written.

== Container object library

This project created a large number of general purpose container objects (classes) for use in higher level code projects. These are all strictly developed under OOA/OOD objectives. They are coded in a mixture of the C and C++ languages. Some of these objects are summarized below:

vecstr	vector strings, stores the data for you
vecpstr	vector of packed strings
vecitem	variable sized items, stores the data for you
vecobj	fixed sized objects, stores the data for you
vechand	you have to store the data yourself
vecint	integers, stores data

recarr	record-array (variable vector) of pointer handles
hdb	you have to store the data yourself
hdbstr	key-value strings, stores data for you
mapstrint	map of strings to integers, stores data
mapstrs	map of a string key with a string value
setstr	set (unique) of strings
osetstr	ordered set (unique) of string
vsetstr	another ordered set (unique) of strings
bits	dynamic (dynamically growable) bit array
varray	dynamically populated array
fsdir	UNIX FS directory functions (object)
dirtree	UNIX FS directory walk functions using an object
wdt	UNIX FS directory walk function similar to 'ftw'
buffer	variable length object continuous buffer management
bufstr	variable length object continuous buffer management (cheapy?)
sbuf	fixed length object continuous buffer management
strmgr	like SBUF but simpler (generally used internally)
storeitem	fixed length object buffer management of separate items
serialbuf	fixed length object buffer management of separate items
outbuf	some whacky thing!
outstore	this is similar (or essentially exactly the same as) BUFSTR
storebuf	fixed length non-object continuous buffer management
strtab	string-table generating object
strstore	string storage object
strpack	a simple string-packing object (lighter weight than STRSTORE)
strop	some (relatively) simple operations on counted strings
netorder	non-object specialized for individual network-order items
stdorder	non-object specialized for individual standard-order items
raqhand	Random-Access-Queue handler (see the code)
fifostr	FIFO object for strings, stores data
fsi	FIFO object for strings, stores data, thread-safe
fifoitem	FIFO object for variable sized items
q	self-relative Q, relocatable-head, thread-safe
piq	pointer Q, relocatable-head, count, magic, thread-safe
aiq	self-relative Q, thread-safe, async, magic, count
plainq	plain self-relative Q, count, and magic
cpq	circular pointer Q (huge in the old days w/ OS stuff!)
pq	pointer Q, relocatable-head, not-circular, count
cq	container Q, relocatable-head, count, magic
ciq	container Q, relocatable-head, count, magic, thread-safe
singlist	a single-link list (not brain-damaged and better than STL)
charq	character Q, relocatable-head, count
chariq	character Q, relocatable-head, count, thread-safe
intiq	integer Q, relocatable-head, count, thread-safe
fmq	file-message queue
pmq	POSIX message queue
psem	POSIX semaphore
csem	Counting-Semaphore (general counting semaphore)
ucsem	UNIX Counting-Semaphore
mailmsgatt	mail attachment management
mailmsg	an object of the general message headers
mailmsgattent	mail-message attachment handing
mailmsgenv	mail-message environment handling
mailmsghdrfold	mail-message header folding
mailmsghdrs	mail-message header management
mailmsghdrval	mail-message header value management
mailmsgheadkey	mail-message header-key matching
mailmsgmatenv	mail-message environment matching
mailmsgmathdr	mail-message header matching
mailmsgstage	mail staging object
mailbox	mail-box object
mailalias	system-wide mail alias DB access
mxalias	user-local alias DB access
msgheaders	a small object to do something on popular message headers
sigign	signal management object
sigblock	signal management object

sighand	signal management object
sigman	signal management object
streamsync	acquire stream character synchronization
dstr	string object
random	UNIXfi System |random(3c)| but made into an object
randomvar	a new high-randomness random number generator (object)
randmwc	random number generator (Multiply W/ Carry)
strlist	manage a STRLIST database file
svcfile	service-table file
kvsfile	key-value file
sysvar	constant database file
pcsconf	constant database file
pcspoll	user-mode PCS polling manager
bstree	Binary-Search-Tree object (slightly more useful than the STL)
obuf	output-buffer (used internally in some point solution things)
listenspec	a high-function network socket and named pipe listener object which handles a variety of listen types; among these, TCP socket, UNIX domain socket, UNIX named pipe (for passed descriptors), and UNIX named pipe for connection (STREAMS connections)
poller	a high-function polling object for use in managing multiplexed I/O

= General purpose utility library (UNIX oriented)

This library provides a large number of general utility subroutines which provide a large number of services for higher level software components. These subroutines can roughly be grouped according to categories as such:

+ UNIX system or user account information retrieval

getenv2	
getenv3	
getpwd	
getpwusername	replacement for 'getpwnam'
getpwlogname	replacement for 'getpwnam'
getlogname	replacement for 'getlogin'
getgroupname	get current user group-name
getutempent	UTMPX functions
getutemterm	UTMPX by terminal-name
getnodename	get the current node-name (will use environment)
getdomainname	get the INET domain-name (will use environment)
getnodedomain	get the node-name and the INET domain (will use environment)
getprojname	get user project-name
getclustername	get the cluster for the current node
getserial	get a serial number from a file-DB
nisdomainname	get the NIS domain name
getax	access the standard UNIX account databases
getcanonical	
getchost	
getchostname	
getcname	
getehostname	
gethe	
gethe1	
getheaddr	
gethostaddr	

+ file-system assistance utility subroutines

getfiledirs	find directories
findfile	find file
findfilepath	find file

findxfile	find eXexecuable file

+ C-style string management subroutines

strcpylc	copy string to lower case
strcpyuc	copy string to upper case
strcpyfc	copy string to folded case
strncpylc	copy to lower case and fill out with NULs
strncpyuc	copy to upper case and fill out with NULs
strncpyfc	copy to folded case and fill out with NULs
strwcpy	copy a maximum length string to another
strwcpylc	to lower case
strwcpyuc	to upper case
strwcpyfc	to folded case
strwcpyrev	to reversed sequence of source
strljoin	join two strings to a destination (w/ destination length)
strnchr	same as 'strchr()' but string has specified length
strnrchr	same as 'strrchr()' but string has specified length
strnpbrk	same as 'strpbrk()' but string has specified length
strwhite	same as 'strpbrk(s," \v\t\r\n")
strnncpy	special copy of strings with length and maximum
strsub	find a substring in a string
strtoken	reentrant version of 'strtok()' (UNIX has something now)
sifield	string-index to field-separator or end
sichr	string-index to character
sisub	string-index to sub-string
sicasesub	string-index to sub-string (case independent)
sibreak	string-index to break characters
sispan	string-index past spanning characters
sibasename	string-index to base-name
siskipwhite	string-index skipping over white-space
sihyphen	string-index to a hyphen ('--')
sialnum	string-index to alpha-numeric character
sialpha	string-index to alpha character
sidquote	string-index to the end of double-quote string
sicite	string-index to a "citation" escape
silbrace	string-index to a left-brace character after white-space
strnlen	get the length of a string w/ a specified maximum
strdcpy<x>	concatenate <x> strings to a counted string buffer
strshrink	shrink off white space from a string
strdirname	find dir-name of a directory file path
strbasename	find base-name of a directory file path
strdomain	INET domain thing
strftime	same as UNIX (now standard) ?
strkeycmp	compare key parts of two strings
strnkeycmp	compare key parts of strings
strnncmp	special compare of strings with length and maximum
strpcmp	prefix comparison of strings
strlead	compare leading parts of strings
strnlead	compare leading parts of strings
snwcpy	copy a counted to a counted string buffer
snwcpylc	copy a counted string to lower case counted string buffer
snwcpyul	copy a counted string to upper case counted string buffer
snwcpyfl	copy a counted string to folded case counted string buffer

sncpy<x>	copy <x> strings to counted string buffer
sncpylc	copy string to lower case counted string buffer
sncpyuc	copy string to upper case counted string buffer
sncpyfc	copy string to folded case counted string buffer

+ UNIX path utility and manipulation

mkpath<x>	make a path from <x> components
mkfname<x>	make a file name from <x> concatenated strings
mkfnamesuf<x>	make a file name from a base and <x> concatenated suffix strings

+ Miscellaneous utility or assistance subroutines

mkpr	make (find) a program root directory
mkbangname	make a "bang" ("n!u") name
mkbestname	make the best real name we can
mkmailname	make a name suitable for use as a mail-address (PCS facility)
mklogid	make a log-id (for logging)
mkplogid	make a log-id (with some difference than 'mklogid')
mkmsgid	make a MSG ID
mkutmpid	make an ID suitable for use in UTMP databases
mktmpfile	make a temporary file
mkjobfile	make a temporary file (suitable as a job-name)
mkdatefile	make a temporary file (suitable for a date thing of some sort)
randlc	Linear Congruent Random Number Generator
mallocstr	malloc()'s a buffer the size of a supplied string
mallocstrn	malloc()'s a buffer the size of a supplied string of MAX length
mallocbuf	malloc()'s a buffer of the size specified

+ Algorithms

bellmanford1	Bellman-Ford algorithm-1
bellmanford2	Bellman-Ford algorithm-2
bfs1	breadth first search
ctwords	Convert-to-Words (also see program NUMCVT)
dfs1	Depth-First-Search algorithm-1
dfs2	Depth-First-Search algorithm-2
dijkstra1	Dijkstra shorted-path-in-graph, algorithm-1
dijkstra2	Dijkstra shorted-path-in-graph, algorithm-2
graph	some graph thing?
minmaxelem	find maximum and minimum in a range over a list-like object
returnstatus	manage return status (I guess for some point solutions)
sort_insertion	an insertion sort (for C++)
sort_merge	a merge sort (for C++)
willAddOver	test for overflow in addition

== Text indexing and search

This project designed both software object modules and utilities to perform text indexing and text searching (on text files or web pages, et cetera). Object modules designed include:

+ textmkind	experimental index creation object
+ txtindexmk	index create object
+ txtindexmks	shared module index create object
+ txtindex	index search object
+ txtindexes	shared module index search object
+ txtindexhdr	index file header
+ textlook	search result verification object

Utility programs designed under this project include:

+ mkkey	make (find) index keys from various text files types
+ mkinv	make index files from generated text keys
+ mkquery	perform a text search on a given index database
+ mktagprint	print out text given a text-search result
+ mkanalysis	perform analysis on text index files

Text manipulation and processing

This project performs text processing or transforming on general text files. It consists of several utility programs. One utility (TEXTCLEAN) performs a variety of text manipulations on generic text files including:

+ extra white-space removal
+ Microsoft character set translation into Latin-1
+ conversion to lower case
+ double space (add intervening blank lines)
+ half space lines (reverse of double-space)
+ extra blank-line removal
+ remove leading white-space
+ remove trailing white-space
+ pad trailing white-space to a right margin column
+ specify disposition of output result

Another utility (TEXTSET) typesets plain text files into the typesetting language TROFF (for further processing). Typesetting options include:

+ point size and vertical spacing (leading)
+ input lines on to typeset on formatted pages
+ starting offset in input file
+ font to use in formatted output
+ top and bottom margin lines in formatted output

Another typesetting utility (COOKIESET) typesets UNIX "cookie" files into source TROFF language.

Other relatively minor utilities perform functions such as:

+ stripdos	strip MS-DOC special characters
+ stripdot	strips lines that lead with a dot character
+ striphighbit	strips the high (8th bit) of bytes
+ stripleading	strips leading white-space
+ stripleadpound	strips leading white-space before pound characters
+ stripopgarb	this is a point utility for striping unnecessary and otherwise obstructive operand information from certain types of program execution trace files
+ lineinvert	line-invert text lines in files
+ linefold	fold long lines in text files
+ mkwords	generic key extraction from text files

Computer language program correctness verification utilities

These utilities (either singly or grouped) provide correctness verification analysis on C and C++ source languages. Analysis outputs optionally consist of fault source line location and counts of the various element openings and closings. These utilities consist of:

+ balanced bracket analysis
+ balanced parenthesis analysis
+ balanced braces analysis
+ balanced comment (opening and closing) analysis
+ balanced single-quote analysis
+ balanced double-quote analysis

Source documentation formatting helper utilities

These utilities serve as helper programs for the typeset processing of source document formatted text files (generally sourced in TROFF, TROFF-MM, or associated source input formats). These utilities are primarily or often embedded within printer utilities or printer facilities. Some of these utilities are:

```
+ gtag          perform citation tag processing for cross referencing of
                citations within a source document
+ mmcite        perform reference citation inclusion from lookup keys in
                source documents
+ referm        perform reference citation inclusion based on citation key words
                such as referenced author and title words
+ imainc        helper utility to include graphic image source files into
                typeset source language documents
```

Numeric conversion utilities

These utilities (TEMP, NUMWORDS, NUMCVT) convert numbers from and to various bases or other representations. Conversions supported include:

+ temperature (and to Fahrenheit and Celsius)
+ from and to all common numeric bases (2, 8, 10, 16, 26)
+ conversion from and to Roman numerals
+ conversion to written out words

Specialized numeric calculator

This is a collection of small calculator utilities which calculate certain combinatorial functions. Calculations performed include:

- factorial (nF)
- exponential (nEk)
- permutations (nPk) , with and without repetitions
- combinations (nCk) , with and without repetitions

Bulletin board management utilities

This project consists of a simple bulletin board facility. The bulletin board facility consists of a user-visible set of hierarchical newsgroups (or boards) that articles can be posted to and read from. A small number of utilities programs (and daemons) are used to run and manage this facility. These utilities are:

```
+ bbpost        high-level article posting
+ rbbpost       daemon for low-level post insertion and also a daemon
                for periodic maintenance and article expiration
+ bbnews        user and administrative management
```

Generic calendar project

This project allows for the access and automatic management of calendar entries. Although this facility can be used for personal use (and system-wide use simultaneously), its primary goal was to allow for access and management of relatively fixed system-wide (or world-wide) historical events. Various databases are maintained for quick calendar entry queries and access. All management and updating of database files is done automatically by the underlying software on-demand (no administrative attention required). The user access facility is CALYEAR.

Most of this system is implemented with OOA objects and OOA loadable shared-object modules. Some of the software object modules are:

+ calcite	citation management object
+ calent	calendar entry object
+ calmgr	calendar manager object
+ calworder	calendar entry duplication detection object
+ calyear	main calendar access object
+ calyears	background loadable shared-object (OOA) module

== UNIX holiday management

This is similar to a calendar entry facility but instead uses an enhanced representation of the UNIX "holidays" database as the source calendars. Unlike normal UNIX systems, multiple years of calendars are possible. The user access utility is HOLIDAY. Some primary software components making up this facility include:

+ holidayer	higher-level multiple calendar manager
+ holidays	lower-level single calendar manager

== UNIX administrative messages

This project implements a flexible messaging facility for local UNIX messages meant to be accessed or viewed by users. The facility has three main sub-facilities. These are the "issue" messaging sub-facility, the "motd" sub-facility, and the "statmsg" sub-facility. Messages are created by administrators (any number) and are displayed to users based primarily on either UNIX group membership or system-login access point or method. Those considered administrators and the number of their message "appearances" is provided through configuration files (listing administrator usernames or IDs and associated messaging directories). The facility finds messages meant to be accessed and viewed by any particular user (based on that user’s credentials or access point or method) and arranges to return those messages for viewing. The "statmsg" sub-facility allows for more flexible and an additional indirection for finding applicable messages to be display and operates on a per administrator basis. Primary software components making up this project include:

+ issue
+ motd
+ statmsg

Utilities provided are:

+ issue
+ motd

== Distributed execution facility

This project provides a remote execution mechanism similar to other Grid-Computing facilities. This project consists of both low-level distributed or remote execution software components as well as the user-level components. This facility operates very similarly to the Berkeley "Remote Shell" (RSH) program except that is spawns the target program to a computer in the cluster best resourced to execute the remote job.

The user-level components consist of:

+ CEX	the primary program to initiate a distributed program execution (used almost the same as RSH)
+ MSU	daemon program to maintain load-average and other statistics about machines in a given cluster
+ MSINFO	administration access and manage of the cluster statistical

database

== Web page helper utilities

These utilities help with the maintenance of web pages. These either run on-demand, as daemons, or either. Some of these are:

+ webcounter	web page counter management
+ mkarticles	creation of web pages (HTML) from text files
+ homepage	creation of simple web local homepage (can also
+ querystring	extraction of the URL query string from web requests

== Source code management utilities

These utilities assist in the creation of Makefiles for program building and compilation. Some of these utilities are:

- makesafe	Makefile helper utility for dependency verification
- makenewer	Makefile helper utility for build installation
- makedate	Makefile helper utility for build date inclusion
- makeinstall	Makefile helper utility for group installation
- makebelow	Makefile helper utility for hierarchical builds

== System logging utilities

These consist of both software components and administrative program utilities. The software components are:

+ logfile	general logging to a file
+ logsys	logging to the UNIX system log device
+ logcons	logging to the UNIX system console

== Generic network dialer project

This project forms a generic networking dial-out facility for clients to connect with servers which are accessed (dialed to over the network) by various protocols other than just contacting a remote TCP port. Dialers for various protocols are implemented as loadable shared-object modules. Dialers are tried and invoked in turn according to dial-out configuration files supplied for each remote service (remote server) configured for the facility. Many remote systems are supported for each remote service and there can be many protocols configured for each remote service. Loadable dialers are tried in an order according the remote service configuration and are loaded and cached for use in implementing specific dialer protocols.

Some of the dialer protocols supported include:

+ UNIX local domain stream-socket
+ UNIX local domain datagram-socket
+ TCP remote host and port (straight)
+ TCP remote host and port w/ the TCPMUX service protocol
+ NLS remote host and port w/ the SysV NLS service protocol
+ FINGER remote host and port w/ FINGER protocol
+ remote UUX (part of UUCP) remote dialer
+ local UUX program dialer

== Network library adaptation helpers for network listening and dialing

A few network listener helper subroutines are available:

+ listentcp	listen on TCP socket

+ listenudp	listen on UDP socket
+ listenpass	listen on local UNIX named-pipe for passed files
+ listenconn	listen on local UNIX named-pipe for connections
+ listenuss	listen on local UNIX domain socket (stream)
+ listenusd	listen on local UNIX domain socket (datagram)

Also, a variety of network dialer subroutines are available. Some of these are:

+ dialtcp	dial out to remote host and port on TCP transport layer
+ dialtcpmux	dial out to remote host and port on TCP transport layer using TCPMUX service protocol
+ dialtcpnls	dial out to remote host and port on TCP transport layer and using the SysV TLI Network Listener Service (NLS) service protocol
+ dialudp	dial out to remote host and port on UDP transport layer
+ dialcprog	dial out to a remote host on the same cluster using load balancing
+ dialfinger	dial out to remote host and port on TCP transport layer using FINGER protocol
+ dialhttp	dial out to remote host and port on TCP transport layer using HTTP protocol
+ dialpass	dial to a named file passing a file descriptor
+ dialprog	dial to a local UNIX program
+ dialticotsord	dial to local or a remove host using SysV Transport Layer Interface (TLI) Connection-Oriented-Orderly release
+ dialticotsordmux	dial out to local or a remote host using SysV Transport Layer Interface (TLI) Connection-Oriented-Orderly release and using the TCPMUX service protocol
+ dialticotsordnls	dial out to local or a remote host using SysV Transport Layer Interface (TLI) Connection-Oriented-Orderly release and using the SysV TLI Network Listener Service (NLS) service protocol
+ dialusd	dial out to a local UNIX domain socket (datagram)
+ dialuss	dial out to a local UNIX domain socket (stream)
+ dialussmux	dial out to a local UNIX domain socket (stream) using the TCPMUX service protocol
+ dialussnls	dial out to a local UNIX domain socket (stream) using the SysV Network Listener Service (NLS) service protocol
+ dialuux	dial out to a remote host using the UNIX UUX facility
+ dialopts	helper to configure dial options for other dialers

== the Korn Shell built-in project

Many built-in commands have been created for use with the Korn Shell. These provide for dramatically increased speed when used in Shell program and particularly when they are inside loops. Over 100 (too numerous to list here) of these built-ins have been created. This is roughly similar to having created the corresponding number of UNIX programs except that these built-ins are dynamically loaded into the Korn Shell on first use for dramatic performance speedups.

== UNIX middleware file-system abstraction

This project makes a number of system services and network protocol ports or

network services available as pseudo-files through means of a middleware
file-system software adaptation layer. In general |open(2)| and |stat(2)| and
many other kernel calls can be interposed upon to provide intercepts for
additional interpretations of certain file path-names and individual file names.
These pseudo-files look and behave very similarly to named FIFOs and mounted
named pipes located in the normal UNIX file-system.

Two main types of service files are emulated. These are rooted files
located under a pseudo-mount point (as if it were a specialized mounted
file-system) and secondly, a so-called floating file which is not under
a mount point and not even necessarily attached to any existing file-system
at all. The first of these basic types takes the form:

```
        /<leading>/<pseudo_mount>/<file-path>[-<arg(s)>]
```

Where the <leading> path components are often (but not always) null, resulting
in the psuedo-mount point residing directly under the root of the system (like
'/proc' for example). As shown, arguments are possible trailing the last
component of the file-path. Examples of this type are:

```
        /proto/tcp/inet/timeserver/daytime
        /proto/tcpmux/inet/someserver/someservice
        /proto/finger/inet/fingerserve/query
        /sys/banner
        /sys/hosts
```

In these above examples, 'proto' and 'sys' were the pseudo-mount points
respectively. Further, 'tcp', 'tcpmux', and 'finger' were protocol
specifications, 'inet' servered an an IP protocol specifier (the default being
IPv4, also as 'inet4'; IPv6 'inet6' being the alternative). These were followed
by literal host names, and then followed by literal service names. Substantially
more complicated forms are possible.

Also, some (roughly) standardized forms such as:

```
        /dev/tcp/[<ipversion>.]<host>/<port>
        /dev/udp/[<ipversion>.]<host>/<port>
```

are also recognized and handled.

The second major form of pseudo-files (floating files) take the form:

```
        <facility><type><file-papth(s)>[-<arg(s)>]
```

where <type> is a special character used by the middleware to locate the
responsible software components (usually always dynamically loaded
shared-objects) responsible for the proper interpretation of the remaining
components. The <facility> component identifies a sub software element that
determines both where additional software components to fulfill the requrest
are located as well as how to interpret the file-name components after the
<type> specifier. Examples of these types of pseudo-files are:

```
        local§daytime
        pcs§name
        tcp¥inet:timeserve:dayime
        tcpmux¥inet:timeserve:dayime
        finger¥inet:timeserve:dayime
```

Substantially more complicated forms are possible, allowing for the specifying
of optional components or arguments for services that take them.

Futher, all UNIX domain stream sockets are handled as regular files without the
normal need for programmers to invoke the various UNIX socket APIs.

Why was this kind of mechanism developed here, and why has it been developed elsewhere in the UNIX world? Because use of files such as these above both greatly reduce the programming overhead of using local and remote (network) services, but they also serve to aggregate the various networking and protocol code into a single place reducing the risks of subtle software errors (as a networking library might also do).