

UNI-REXX

ED SPIRE
Workstation Group

uni-REXX

Rexx for Unix

Rexx Symposium

May, 1993

La Jolla, California

The Workstation Group Rosemont, Illinois

iX Corporation Chicago, Illinois

Rexx Symposium -May, 1993
La Jolla, California

The Workstation Group
Rosemont, Illinois

Recent uni-REXX support

TRL-II Issues

stdin:, stdout:, stderr: named streams

variable **sublists** in DROP and EXPOSE

3rd parm of **value()**

"**b**" date format

several obscure conformance issues

Recent uni-REXX support

irxstk, irxexcom via IPC, not linking.

EXECIO

execio {lines | *} {DISKR | DISKW} file (linenum) (((options))

– DISKR options: avoid, find, locate, fifo, finis, lifo, margins,
notype, stem, strip, var, zone

– DISKW options: case, finis, margins, **notype**, string, strip,
stem, var

GLOBALV

globalv select group

(select group) {set | sets | **setp**} name1 value1 . . . nameN valueN

{**setl** | setls | **setsl** | setlp | setpl} name value

{put | puts | putp | list | get | stack} name1 . . . nameN

select group purge

purge

grplist

grpstac k

Recent uni-REXX support

procedure expose for external procedures

Exposure of standard Unix system programming interfaces to REXX:

-accept	-bind	_close
_closedir	_connect	_errno
-exit	-fork	_geteuid
_gethostbyname	_gethostid	_gethostname
_getpid	_getppid	_getservbyname
_getuid	-kill	-listen-opendir
_readdir	_recv	_regex
-send	_setsid	_sleep
-socket	-stat	_sys_errlist
-truncate	_umask	-wait
_waitpid		

```

/*
** Copyright (C) iX Corporation 1993. All rights reserved.
**
** Module =
**
** syserr.rex
**
** Abstract =
**
** System error routine. Display the error number and message for
** a system error.
**
** History =
**
** 07-May-93 Added this comment
**
** Possible future enhancements =
**
*/
procedure expose sial
/*
* display error messages
*/
say "Error in" arg(2) "line" sigl
say arg( 1) "error" _errno():"_sys_errlist(_errno())
/*
* exit for good
*/
call _exit(1)

```

```

/*
** Copyright (C) iX Corporation 1993. All rights reserved.
**
** Module =
**
** sendbuf.rex
**
** Abstract =
**
** Send a buffer in a length prefixed packet
**
** History =
**
** 07-May-93 Added this comment
**
_ ** Possible future enhancements =
**
*/
sendbuf:procedure
parse arg socket, buffer
buff erlength = right(length(buffer), 4, '0')
bufferlength = right(bufferlength + length(bufferlength), 4, '0')
call send socket, bufferlength | buffer, ""
if sendrc c 0 then call sockerr "send"
return

```

```
/*  
** Copyright (C) iX Corporation 1993. All rights reserved.  
**
```

```
** Module =  
**
```

```
** recvbuf.rex  
**
```

```
** Abstract =  
**
```

```
** Receive a buffer in a length prefixed packet  
**
```

```
** History =  
**
```

```
** 07-May-93 Added this comment  
**
```

```
** Possible future enhancements =  
**
```

```
*/
```

```
recvbuf:procedure
```

```
parse arg socket
```

```
rcvrc = recv(socket, "bufferlength", 4, MSG PEEK)
```

```
if rcvrc < 0 then call sockerr "recv"
```

```
rcvrc = recv(socket, "buffer", bufferlength, "")
```

```
if rcvrc < 0 then call sockerr "recv"
```

```
return substr(buffer, 5)
```


Interprocess Communication in uni-REXX

architecture

- Before issuing a command to the OS, the language processor opens a socket and places the address of the socket in an environment variable.**
- The Rexx API library used for external commands contains routines that communicate with the language processor via that socket.**
- Original Rexx API library remains available for embedded applications, allowing direct access to the language processor APIs.**

Interprocess Communication in uni-REXX

Applications to date:

- **globalv**
- execio
- rxsql (oracle)

Performance Considerations

C library bif's (sockets, multitasking, etc.) allow for similar coding in REXX rather than **C**.

```

/*
** Copyright (C) iX Corporation 1993. All rights reserved.
**
** Module =
**
** iserver.rex
**
** Abstract =
**
** Demonstrate Open-REXX UNIX system interfaces with a client
** server implementation. This routine is the server. It must be run
** as root. Only one copy should run at once, or unusual things
** may happen. To stop it, simply kill the PID displayed at start-up.
**
** Note: The service name rexxinet must be in /etc/services.
**
** This server simply accepts a file mask and returns all the files
** that match it in the current directory. The server signals it's
** done by sending "<end>".
**
** The client and server may be anywhere on the same network.
**
** History =
**
** 07-May-93 Added this comment
**
** Possible future enhancements =
**
*/

```

```

/*
 * note the program name
 */
parse source. . pgmname .
/*
 * create the client process
 */
forkrc = _fork()
if forkrc < 0 then call syseri- "fork", pgrname
/*
 * the parent now exits, leaving only the child
 */
if forkrc <> 0 then
  do
    say "REXX daemon started: PID =" forkrc
    exit
  end
/*
 *
 */

```

```

setidrc = _setsid()
if setidrc < 0 then call syserr "setid", pgmname
call _umask(0)
/*
 * open the server internet socket
*/
socket0 = _socket(AF_INET, SOCK_STREAM, 0)
if socket0 < 0 then call syserr "socket", pgmname
/*
 * get the server structure
*/
call _getservbyname("rex/inet", "tcp", "server.")
/*
 * initialize the internet socket address structure
*/
inetsocket.sa_family = AF_INET
inetsocket.sin_addr.s_addr = INADDR_ANY
inetsocket.sin_port = server.s_port
/*
 * bind the socket to the port
*/
bindrc = _bind(socket0, "inetsocket.")
if bindrc < 0 then call syserr "bind", pgmname
/*
 * listen for connections
*/
listenrc = _listen(socket0, 5)
if listenrc < 0 then call syserr "listen", pgmname

```

```

/*
 * process client connections as they appear
 */
do forever
    /*
     * accept the client connection
     */
    socket1 = _accept(socket0, "inetsocket.", 0)
    if socket1 < 0 then call syserr "accept", pgmname
    /*
     * get the client file mask
     */
    mask = recvbuf(socket1)
    /*
     * convert the file mask to a regular expression
     */
    regex = sh2reg(mask)
    /*
     * open the current directory
     */
    dir = _opendir(".")
    if dir < 0 then call syserr "opendir", pgmname

```

```

/*
 * send each matching file name back to the client
 */
do forever
  /*
   * get the next file
   */
  currentfile = _readdir(dir)
  /*
   * if it's the last file, we're done
   */
  if currentfile == "" then
    leave
  /*
   * if the file matches the pattern, send it to the client
   */
  if _regex(regex, currentfile) = 1 then
    call sendbuf socket 1, currentfile
  end
end

/*
 * send the end-of transaction indicator
 */
call sendbuf socket 1, "<end>"
/*
 * close the client connection
 */
closerc = _close(socket1)
if closerc < 0 then call syserr "close", pgmname
end

/*
 * close the accepting connection
 */
closerc = _close(socket0)
if closerc < 0 then call syserr "close", pgmname

```

```

/*
** Copyright (C) iX Corporation 1993. All rights reserved.
**
** Module =
**
** iclient.rex
**
** Abstract =
**
** Demonstrate Open-REXX UNIX system interfaces with a client
** server implementation. This routine is a client. It connects with
** an internet service named "rexxinet" and sends it a file mask.
** The server should then respond with each file in its current dir,
** that it matches. A buffer containing "<end>" signals completion.
**
** Note: The service name rexxinet must be in etc /services.
**
** The client and server may be anywhere on the same network,
**
** History =
**
** 07-May-93 Added this comment
**
** Possible future enhancements =
**
*/

```



```

mask = "*.rex"
/*
 * note the program name
 */
parse source. . pgmname .
/*
 * get the host structure
 */
call_gethostbyname(_gethostname(),"ph.")
/*
 * get the server structure
 */
call_getservbyname("rexinet", "tcp", "ps. ")
/*
 * initialize the internet socket address structure
 */
sin.sa_family = ph.h_addrtype
sin.sin_addr = ph.h_addr
sin.sin_port = ps.s_port;
/*
 * create the internet socket
 */
socket = _socket(AF_INET, SOCK-STREAM, 0)
if socket < 0 then call syserr "socket", pgmname
/*
 * connect to the server
 */
connectrc = _connect(socket,"sin.")
if connectrc c 0 then call syserr "connect", pgmname
/*
 * send the file mask
 */
call sendbuf socket, mask

```

```

/*
 * get the directory line
 */
say "Response from server for" mask
do forever
    /*
     * get the buffer from the client
     */
    buffer = recvbuf(socket)
    /*
     * if it's the end of transaction indicator, we're done
     */
    if buffer == "<end>" then
        leave
    /*
     * display the file
     */
    say buffer
end
/*
 * close the internet socket
 */
closerc = _close(socket)
if closerc < 0 then call syserr "close", pgmname

```

Market Acceptance

Last year, many research purchases,
few pilot projects

This year, many pilot projects, some
production implementations

Unbundling Rexx from **it's** embedded
applications has helped cost justify it's
acquisition by commercial accounts