DQS 3.0 Library Interface

Supercomputer Computations Research Institute
Florida State University
Tallahassee, Florida

Thomas P. Green

Contents

I Reference Manual

Part I

Reference Manual

Chapter 1

Commands

1.1 dqs_add_complex(c)

File dqs_add_del.c

int dqs_add_complex(c)

dqs_list_type *c;

dqs_add_complex - adds a job to the Complex_list and to Complex_hash
if the invoking process is the qmaster the complex list is spooled
to disk.
returns:
-1 if complex already exists;
0 if successful
1 on error

1.2 dqs_add_host(h)

File dqs_add_del.c

int dqs_add_host(h)

char *h;

dqs_add_host - adds a host to the Host_list and to Host_hash
if the invoking process is the qmaster the host list is
spooled to disk
returns
-1 if host already exists or other error;
0 if successfull;
1 if invalid hostname.

1.3 dqs_add_job(j)

File dqs_add_del.c
int dqs_add_job(j)
dqs_job_type *j;
dqs_add_job - adds a job to the Job_list and to Job_hash
if the invoking process is the qmaster the job is spooled
to disk.
returns:
-1 if job already exist;
0 if successful
1 on error

1.4 dqs_add_queue(q)

File dqs_add_del.c
int dqs_add_queue(q)
dqs_queue_type *q;
dqs_add_queue - adds a queue to the Queue_list and to Queue_hash.
if the respective host is not in Host_list/Host_hash calls dqs_add_host()
if the invoking process is the qmaster the queue is spooled to disk.
returns:
-1 if queue already exists or other error;
0 if successful;
1 if invalid host associated with queue.

1.5 dqs_alarmclock()

File dqs_sig_handlers.c
void dqs_alarmclock()

1.6 dqs_args2list(argv,

File dqs_list.c
dqs_list_type *dqs_args2list(argv,
argc)
dqs_args2list - converts an argument list to linked list

1.7 dqs_atoi(int_str)

File dqs_utility.c

int dqs_atoi(int_str)

char *int_str;

1.8 dqs_build_qcomplex_str(qcomplex)

File dqs_job_exit.c

char *dqs_build_qcomplex_str(qcomplex)

dqs_list_type *qcomplex;

dqs_build_qcomplex_str - construct the string equilavent of a resource list
Returns: "none" if listel_ptr is NULL
otherwise, the string equilavent of a resource list

1.9 dqs_c_dqs_execd(sfd,request_head)

File dqs_c_dqs_execd.c

void dqs_c_dqs_execd(sfd,request_head)

int sfd;
dqs_list_type **request_head;

dqs_c_dqs_execd - is used solely by the qmaster and handles requests forwarded from the dqs_execd.
The most commonly used service is that of the dqs_execd sending in load average information.
Secondly is that of the dqs_execd sending in statistics of reaped jobs.
Other interfaces include the dqs_execd informing the qmaster that it has bee restarted.

1.10 dqs_c_qconf(sfd,request_head)

File dqs_c_qconf.c

void dqs_c_qconf(sfd,request_head)

int sfd;
dqs_list_type **request_head;

dqs_c_qconf is used solely by the qmaster and provides services provides services to requests forwarded by the qconf ancillary.

1.11 dqs_c_qdel(sfd,request_head)

File dqs_c_qdel.c

void dqs_c_qdel(sfd,request_head)
int sfd;
dqs_list_type **request_head;
dqs_c_qdel- is used solely by the qmaster and provides services to the qdel ancillary.

1.12 dqs_c_qmaster(sfd,request_head)

File dqs_c_qmaster.c

void dqs_c_qmaster(sfd,request_head)
int sfd;
dqs_list_type **request_head;
dqs_c_qmaster is used solely by the dqs_execd and provides services to requests forwarded by the qmaster.
Some of the services provided include job execution, queue suspension, and job signaling.

1.13 dqs_c_qmod(sfd,request_head)

File dqs_c_qmod.c

void dqs_c_qmod(sfd,request_head)
int sfd;
dqs_list_type **request_head;
dqs_c_qmod is used solely by the qmaster and provides services to the qmod ancillary.

1.14 dqs_c_qstat(sfd,request_head)

File dqs_c_qstat.c

void dqs_c_qstat(sfd,request_head)
int sfd;
dqs_list_type **request_head;
dqs_c_qstat - is utilized solely by the qmaster and provides service to the qstat ancillary.

1.15 dqs_c_qsub(sfd,request_head)

File dqs_c_qsub.c
void dqs_c_qsub(sfd,request_head)
int sfd;
dqs_list_type **request_head;
dqs_c_qsub - is used solely by the qmaster and provides services
to the qsub ancillary.

1.16 dqs_chdir(path)

File dqs_setup.c
void dqs_chdir(path)
char *path;

1.17 dqs_ck_to_do_list()

File dqs_ck_to_do_list.c
void dqs_ck_to_do_list()
This module is used by the qmaster and dqs_execd.
If the invoker is the qmaster, this function provides self reauthentication
services - if so configured.
If invoked by the dqs_execd, this routine also checks for jobs which
have surpassed there wall clock time limits, and job reauthentication.

1.18 dqs_clear_granted_resources(job)

File dqs_job_exit.c
void dqs_clear_granted_resources(job)
dqs_job_type *job;
dqs_clear_granted_resources - clears granted resources which have not been activated

1.19 dqs_clear_suitability()

File dqs_select_queue.c
void dqs_clear_suitability()
* dqs_clear_suitability - clears the "suitability field on all queues.

1.20 dqs_close_fds()

File dqs_utility.c
void dqs_close_fds()

1.21 dqs_close_sfd(sfd)

File dqs_send_receive.c
void dqs_close_sfd(sfd)
int sfd;

1.22 dqs_commit_job(job)

File dqs_schedule.c
void dqs_commit_job(job)
dqs_job_type *job;
dqs_commit_job - commits that a job has been started
several things need to be done,
mark granted resources as "";
tag respective queues;
head of granted_destination_id marked as "master"
spool dirty queues to disk

1.23 dqs_complex_el_suitable(req_lp,c_lp)

File dqs_select_queue.c
int dqs_complex_el_suitable(req_lp,c_lp)
dqs_complex_el_suitable - determines whether a complex element actually
fits a request.
returns 0 on failure; 1 on success

1.24 dqs_concat_buf(script_ptr,str,script_size)

File dqs_utility.c
char *dqs_concat_buf(script_ptr,str,script_size)
char *script_ptr;
char *str;
u_long32 *script_size;

1.25 dqs_ctime(i)

File dqs_utility.c
char *dqs_ctime(i)
time_t i;

1.26 dqs_daemonize()

File dqs_daemonize.c
void dqs_daemonize()
dqs_daemonize - throws ourself into the background and dissassociates
from any controlling ttys

1.27 dqs_decrypt_token(token)

File dqs_reauth.c
void dqs_decrypt_token(token)
char *token;

1.28 dqs_del_job(job_name)

File dqs_add_del.c
int dqs_del_job(job_name)
char *job_name;
dqs_del_job - deletes a job from the Job_list and Job_hash
and unlinks associated files.
returns:
-1 on error;
0 if successful;

1.29 dqs_del_str0(head,str)

File dqs_list.c
dqs_list_type *dqs_del_str0(head,str)
dqs_list_type *head;
char *str;
dqs_del_str0 - delete list entry based on str0 field

1.30 dqs_delete_head(head)

File dqs_list.c
dqs_list_type *dqs_delete_head(head)
dqs_delete_head - deletes first entry in linked list

1.31 dqs_dequalify_hostname(old_host_str,host_str)

File dqs_utility.c
char *dqs_dequalify_hostname(old_host_str,host_str)
char *old_host_str;
char *host_str;

1.32 dqs_dequalify_path(old_path_str,path_str)

File dqs_utility.c
char *dqs_dequalify_path(old_path_str,path_str)
char *old_path_str;
char *path_str;

1.33 dqs_do_log(log_level,err_str)

File dqs_log.c
void dqs_do_log(log_level,err_str)
int log_level;
char *err_str;

1.34 dqs_do_reauth(job)

File dqs_reauth.c
void dqs_do_reauth(job)
dqs_job_type *job;

1.35 dqs_do_schedule(job)

File dqs_schedule.c
int dqs_do_schedule(job)
dqs_job_type *job;

1.36 dqs_domkdir(path_,owner,group,fmode)

File dqs_setup.c
void dqs_domkdir(path_,owner,group,fmode)
char *path_;
int owner;
int group;
mode_t fmode;

1.37 dqs_edit(fname)

File dqs_parse_qconf.c
int dqs_edit(fname)
char *fname;

1.38 dqs_encrypt_token(token)

File dqs_reauth.c
void dqs_encrypt_token(token)
char *token;

1.39 dqs_exec_job(listel_ptr)

File dqs_exec_job.c
int dqs_exec_job(listel_ptr)
dqs_list_type *listel_ptr;
dqs_exec_job - is used soley by the dqs_execd.
this routine is responsible for setting up the process environment
prior to an execl.

1.40 dqs_execd_deliver_signal(sig,lp)

File dqs_c_qmaster.c
void dqs_execd_deliver_signal(sig,lp)
u_long32 sig;
dqs_list_type *lp;

1.41 dqs_fgetstring(fp,str,size)

File dqs_utility.c

int dqs_fgetstring(fp,str,size)
FILE *fp;
char *str;
int size;

1.42 dqs_fill_queue(job,qname,where)

File dqs_schedule.c

int dqs_fill_queue(job,qname,where)
dqs_job_type *job;
char *qname;
int where;

1.43 dqs_free(cp)

File dqs_utility.c

char *dqs_free(cp)
char *cp;

1.44 dqs_free_conf(c)

File dqs_free.c

dqs_conf_type *dqs_free_conf(c)
dqs_conf_type *c;
dqs_free_conf - delete a dqs_conf_type structure.

1.45 dqs_free_job(j)

File dqs_free.c

dqs_job_type *dqs_free_job(j)
dqs_job_type *j;
dqs_free_job - delete a dqs_job_type structure.

1.46 dqs_free_list(sp)

File dqs_free.c
dqs_list_type *dqs_free_list(sp)
dqs_list_type *sp;
dqs_free_list - delete linked list dynamic memory

1.47 dqs_free_me(m)

File dqs_free.c
dqs_me_type *dqs_free_me(m)
dqs_me_type *m;
dqs_free_me - delete a dqs_me_type structure.

1.48 dqs_free_queue(q)

File dqs_free.c
dqs_queue_type *dqs_free_queue(q)
dqs_queue_type *q;
dqs_free_queue - delete a dqs_queue_type structure.

1.49 dqs_free_rusage(r)

File dqs_free.c
dqs_rusage_type *dqs_free_rusage(r)
dqs_rusage_type *r;
dqs_free_rusage - delete a dqs_rusage_type structure.

1.50 dqs_get_ack(sfd)

File dqs_send_receive.c
int dqs_get_ack(sfd)
int sfd;

1.51 dqs_get_dirents(path)

File dqs_setup.c
dqs_list_type *dqs_get_dirents(path)
char *path;

1.52 dqs_get_env_list(head,envp)

File dqs_parse.c

```
void *dqs_get_env_list(head,envp)
dqs_list_type **head;
char **envp;
```

str0 = env_name
str1 = env_description
str3 = tempoary
int3 = tempoary

1.53 dqs_get_gmt()

File dqs_utility.c

```
u_long32 dqs_get_gmt()
```

1.54 dqs_get_job_number()

File dqs_utility.c

```
u_long32 dqs_get_job_number()
```

1.55 dqs_get_key(key)

File dqs_reauth.c

```
int dqs_get_key(key)
char *key;
```

1.56 dqs_get_list(sfd,head)

File dqs_send_receive.c

```
int dqs_get_list(sfd,head)
int sfd;
dqs_list_type **head;
```

dqs_get_list_ - should only be called from dqs_get_list
returns 0 on success
returns -1 on failure

1.57 dqs_get_loadavg()

File dqs_load_avg.c

int dqs_get_loadavg()

1.58 dqs_get_pass()

File dqs_reauth.c

dqs_list_type *dqs_get_pass()

1.59 dqs_get_passwd_info(job)

File dqs_reauth.c

void dqs_get_passwd_info(job)

```
dqs_job_type *job;
```

1.60 dqs_get_path(lp,job_name,job_number,type)

File dqs_utility.c

char *dqs_get_path(lp,job_name,job_number,type)

```
dqs_list_type *lp;
char *job_name;
u_long32 job_number;
int type;
```

1.61 dqs_get_tid()

File dqs_tid.c

dqs_list_type *dqs_get_tid()

1.62 dqs_getme(dqs_formal_prog_name,qualified_prog_name)

File dqs_setup.c

void dqs_getme(dqs_formal_prog_name,qualified_prog_name)

```
u_long32 dqs_formal_prog_name;
char *qualified_prog_name;
```

1.63 dqs_give_job(job)

File dqs_schedule.c

void dqs_give_job(job)

dqs_job_type *job;

1.64 dqs_hash(str)

File dqs_hash.c

int dqs_hash(str)

char *str;

1.65 dqs_hash_add(str,h_table)

File dqs_hash.c

int dqs_hash_add(str,h_table)

char *str;

dqs_hash_type *h_table;

1.66 dqs_hash_add_queue(q)

File dqs_hash.c

int dqs_hash_add_queue(q)

dqs_queue_type *q;

dqs_hash_add_queue - adds a queue to Host_hash
returns -1 if queue already exist;
0 if successful
1 other recoverable error
abort() non-recoverable error

1.67 dqs_hash_del(str,h_table)

File dqs_hash.c

int dqs_hash_del(str,h_table)

char *str;

dqs_hash_type *h_table;

1.68 dqs_hash_init()

File dqs_hash.c

dqs_hash_type *dqs_hash_init()

1.69 dqs_hash_lookup(str,h_table)

File dqs_hash.c

dqs_hash_type *dqs_hash_lookup(str,h_table)

```
char *str;
dqs_hash_type *h_table;
```

1.70 dqs_indent()

File dqs_log.c

void dqs_indent()

```
dqs_indent()
```

1.71 dqs_insert(field,

File dqs_list.c

dqs_list_type *dqs_insert(field,

```
direction, head, element)
u_long32 field;
int direction;
dqs_list_type *head;
dqs_list_type *element;
```

dqs_insert - insert entry into linked list.
entry: field = TYPE, pick insertion point based on type field
= INT0, pick insertion point based on int0 field
= INT1, pick insertion point based on int1 field
= STR0, pick insertion point based on str0 field
= STR0, pick insertion point based on str1 field
direction = HEAD, insert entry at list head
= TAIL, insert entry at list tail
= ASCENDING, do ascending order insertion
= DESCENDING, do descending order insertion
head = pointer to list head
element = pointer to new list entry
exit: returns pointer to (updated) list head

1.72 dqs_insert_sort_queue_x_load(element,head)

File dqs_list.c
dqs_list_type *dqs_insert_sort_queue_x_load(element,head)
dqs_list_type *element;
dqs_list_type *head;
dqs_insert_sort_queue_x_load - used by dqs_sort_queues()

1.73 dqs_insert_sort_queue_x_seq_no(element,head)

File dqs_list.c
dqs_list_type *dqs_insert_sort_queue_x_seq_no(element,head)
dqs_list_type *element;
dqs_list_type *head;
dqs_insert_sort_queue_x_seq_no - used by dqs_sort_queues()

1.74 dqs_is_int(str)

File dqs_utility.c
int dqs_is_int(str)
char *str;

1.75 dqs_is_str(str)

File dqs_utility.c
int dqs_is_str(str)
char *str;

1.76 dqs_job_exit(request_list)

File dqs_job_exit.c
void dqs_job_exit(request_list)
dqs_list_type *request_list;

1.77 dqs_job_owner(user_name,job_name)

File dqs_sec.c
int dqs_job_owner(user_name,job_name)
char *user_name;
char *job_name;

Determines if the user is an owner of a job.
Note: a manager/operator is implicitly an "owner".
returns : -1 if unable to locate the job.
0 if a valid owner
1 not a valid owner

1.78 dqs_kill(pid,dqs_signal)

File dqs_c_qmaster.c
int dqs_kill(pid,dqs_signal)
int pid;
u_long32 dqs_signal;

1.79 dqs_length_of_list(head)

File dqs_list.c
int dqs_length_of_list(head)
dqs_list_type *head;
dqs_length_of_list - compute number of entries on linked list

1.80 dqs_locate_acl(cp)

File dqs_hash.c
dqs_hash_type *dqs_locate_acl(cp)
char *cp;

1.81 dqs_locate_complex(cp)

File dqs_hash.c
dqs_list_type *dqs_locate_complex(cp)
char *cp;

1.82 dqs_locate_host(cp)

File dqs_hash.c
dqs_hash_type *dqs_locate_host(cp)
char *cp;

1.83 dqs_locate_int0(head,int0)

File dqs_list.c

dqs_list_type *dqs_locate_int0(head,int0)

dqs_list_type *head;

u_long32 int0;

dqs_locate_int0 - locate list entry by int0 field

1.84 dqs_locate_job(cp)

File dqs_hash.c

dqs_job_type *dqs_locate_job(cp)

char *cp;

1.85 dqs_locate_manager(cp)

File dqs_hash.c

dqs_hash_type *dqs_locate_manager(cp)

char *cp;

1.86 dqs_locate_operator(cp)

File dqs_hash.c

dqs_hash_type *dqs_locate_operator(cp)

char *cp;

1.87 dqs_locate_queue(cp)

File dqs_hash.c

dqs_queue_type *dqs_locate_queue(cp)

char *cp;

1.88 dqs_locate_str0(head,str0)

File dqs_list.c

dqs_list_type *dqs_locate_str0(head,str0)

dqs_list_type *head;

char *str0;

dqs_locate_str0 - locate list entry by str0 field

1.89 dqs_locate_tid(tid)

File dqs_tid.c

```
dqs_list_type *dqs_locate_tid(tid)
dqs_list_type *tid;
```

1.90 dqs_log(log_level,mesg,file__,func__,line__)

File dqs_log.c

```
void dqs_log(log_level,mesg,file__,func__,line__)
int log_level;
char *mesg;
char *file__;
char *func__;
int line__;
```

DO NOT USE DQS_ASSERT OR LOGGING MACROS IN THIS ROUTINE!
THIS MODULE MUST ACT STAND-ALONE!

1.91 dqs_log_dusage(d,q)

File dqs_job_exit.c

```
void dqs_log_dusage(d,q)
dqs_rusage_type *d;
dqs_queue_type *q;
```

1.92 dqs_log_pid()

File dqs_utility.c

```
void dqs_log_pid()
```

1.93 dqs_mail(user_list,subj,buf)

File dqs_mail.c

```
void dqs_mail(user_list,subj,buf)
dqs_list_type *user_list;
char *subj;
char *buf;
```

1.94 dqs_make_tmpdir(qconf,job)

File dqs_tmpdir.c
char *dqs_make_tmpdir(qconf,job)
dqs_queue_type *qconf;
dqs_job_type *job;

1.95 dqs_malloc(size)

File dqs_utility.c
char *dqs_malloc(size)
int size;

1.96 dqs_manager(cp)

File dqs_sec.c
int dqs_manager(cp)
char *cp;

1.97 dqs_mark_clean(job)

File dqs_schedule.c
void dqs_mark_clean(job)
dqs_job_type *job;

1.98 dqs_mark_unheard(host)

File dqs_utility.c
void dqs_mark_unheard(host)
char *host;

1.99 dqs_mini_setup(prog_name)

File dqs_setup.c
void dqs_mini_setup(prog_name)
char *prog_name;

1.100 dqs_mkdir(path,owner,group,fmode)

File dqs_setup.c

void dqs_mkdir(path,owner,group,fmode)
char *path;
int owner;
int group;
int fmode;

1.101 dqs_my_domain(h)

File dqs_sec.c

int dqs_my_domain(h)
char *h;

1.102 dqs_next_is_an_opt(ptr)

File dqs_parse_qconf.c

int dqs_next_is_an_opt(ptr)
dqs_list_type *ptr;

1.103 dqs_open_tcp

File dqs_io.c

int dqs_open_tcp
(host, port, service, reserved_port)
char *host;
int port;
char *service;
int reserved_port;

dqs_open_tcp - open TCP connection.
entry: host = pointer to hostname
port = 0, get port number from service entry
¿ 0, use specified port number
service = pointer to service name
reserved_port = 0, open socket using non-privileged port
!= 0, open socket using only privileged port
exit: returns file descriptor to opened socket

1.104 dqs_operator(cp)

File dqs_sec.c

int dqs_operator(cp)

```
char *cp;
```

Note: a manager is implicitly an "operator".

1.105 dqs_owner(cp,lp)

File dqs_sec.c

int dqs_owner(cp,lp)

```
char *cp;
dqs_list_type *lp;
```

Note: a manager/operator is implicitly an "owner".

1.106 dqs_pack_conf(head_ptr,cur_ptr,conf,bytes_written,mem_size)

File dqs_pack.c

void dqs_pack_conf(head_ptr,cur_ptr,conf,bytes_written,mem_size)

```
char **head_ptr;
char **cur_ptr;
dqs_conf_type *conf;
int *bytes_written;
int *mem_size;
```

1.107 dqs_pack_header(buf_ptr,nibbles,bytes_packed)

File dqs_pack.c

void dqs_pack_header(buf_ptr,nibbles,bytes_packed)

```
char *buf_ptr;
int nibbles;
int bytes_packed;
```

1.108 dqs_pack_job(head_ptr,cur_ptr,job,bytes_written,mem_size)

File dqs_pack.c

void dqs_pack_job(head_ptr,cur_ptr,job,bytes_written,mem_size)

```
char **head_ptr;
char **cur_ptr;
dqs_job_type *job;
int *bytes_written;
int *mem_size;
```

1.109 dqs_pack_list(head_ptr,cur_ptr,list_head,bytes_written,mem_size,qty)

File dqs_pack.c

void dqs_pack_list(head_ptr,cur_ptr,list_head,bytes_written,mem_size,qty)

```
char **head_ptr;
char **cur_ptr;
dqs_list_type *list_head;
int *bytes_written;
int *mem_size;
int qty;
```

returns char *

returns NULL on error

1.110 dqs_pack_me(head_ptr,cur_ptr,me,bytes_written,mem_size)

File dqs_pack.c

void dqs_pack_me(head_ptr,cur_ptr,me,bytes_written,mem_size)

```
char **head_ptr;
char **cur_ptr;
dqs_me_type *me;
int *bytes_written;
int *mem_size;
```

1.111 dqs_pack_queue(head_ptr,cur_ptr,queue,bytes_written,mem_size)

File dqs_pack.c

```
void dqs_pack_queue(head_ptr,cur_ptr,queue,bytes_written,mem_size)
char **head_ptr;
char **cur_ptr;
dqs_queue_type *queue;
int *bytes_written;
int *mem_size;
```

1.112 dqs_pack_rusage(head_ptr,cur_ptr,rusage,bytes_written,mem_size)

File dqs_pack.c

```
void dqs_pack_rusage(head_ptr,cur_ptr,rusage,bytes_written,mem_size)
char **head_ptr;
char **cur_ptr;
dqs_rusage_type *rusage;
int *bytes_written;
int *mem_size;
```

1.113 dqs_pack_token(head_ptr,cur_ptr,cp,bytes_written,mem_size,qty)

File dqs_pack.c

```
void dqs_pack_token(head_ptr,cur_ptr,cp,bytes_written,mem_size,qty)
char **head_ptr;
char **cur_ptr;
char *cp;
int *bytes_written;
int *mem_size;
int qty;
```

returns char *

returns NULL on error

1.114 dqs_pack_token_list(head_ptr,cur_ptr,list_head,bytes_written,mem_size,qty)

File dqs_pack.c

void dqs_pack_token_list(head_ptr,cur_ptr,list_head,bytes_written,mem_size,qty)

char **head_ptr;
char **cur_ptr;
dqs_list_type *list_head;
int *bytes_written;
int *mem_size;
int qty;

returns char *
returns NULL on error

1.115 dqs_parse_account_list(head,account_str)

File dqs_parse.c

void dqs_parse_account_list(head,account_str)

dqs_list_type **head;
char *account_str;

str0 - account
str1 - cell
str3 - temporary
int3 - temporary

1.116 dqs_parse_acl_list(head,users,list_names,opt)

File dqs_parse_qconf.c

void dqs_parse_acl_list(head,users,list_names,opt)

dqs_list_type **head;
char *users;
char *list_names;
int opt;

1.117 dqs_parse_checkpoint_attr(attr_str)

File dqs_parse.c

int dqs_parse_checkpoint_attr(attr_str)

char *attr_str;

1.118 dqs_parse_checkpoint_interval(minutes_str)

File dqs_parse.c
int dqs_parse_checkpoint_interval(minutes_str)
char *minutes_str;

1.119 dqs_parse_date_time(date_str)

File dqs_parse.c
u_long dqs_parse_date_time(date_str)
char *date_str;
-a date_time
"touch -t" format
[[CC]YY]MMDDhhmm[.SS]

1.120 dqs_parse_destination_id_list(head,cp,opt)

File dqs_parse_qconf.c
void *dqs_parse_destination_id_list(head,cp,opt)
dqs_list_type **head;
char *cp;
int opt;

1.121 dqs_parse_destination_identifier_list(head,dest_str)

File dqs_parse.c
void dqs_parse_destination_identifier_list(head,dest_str)
dqs_list_type **head;
char *dest_str;
destination_identifier "queue" "@cell" "queue@cell"
str0 - queue
str1 - cell
str3 - temporary
int3 - temporary

1.122 dqs_parse_hold_list(hold_str)

File dqs_parse.c
int dqs_parse_hold_list(hold_str)
char *hold_str;

1.123 dqs_parse_job(head,envp)

File dqs_parse.c

dqs_job_type *dqs_parse_job(head,envp)
dqs_list_type *head;
char **envp;

1.124 dqs_parse_job_identifier_list(head,job_str)

File dqs_parse.c

void dqs_parse_job_identifier_list(head,job_str)
dqs_list_type **head;
char *job_str;

job_identifier sequence_number[.server_name][@server]
str0 - sequence_number_str
str0 - cell
str3 - temporary
int0 - sequence_number
int3 - temporary

1.125 dqs_parse_keep_list(keep_str)

File dqs_parse.c

int dqs_parse_keep_list(keep_str)
char *keep_str;

1.126 dqs_parse_mail_list(head,mail_str)

File dqs_parse.c

void dqs_parse_mail_list(head,mail_str)
dqs_list_type **head;
char *mail_str;

user[@host][,user[@host],...]
str0 - user
str1 - host
str3 - temporary
int1 - temporary

1.127 dqs_parse_mail_options(mail_str)

File dqs_parse.c
int dqs_parse_mail_options(mail_str)
char *mail_str;

1.128 dqs_parse_name_list(head,cp,opt)

File dqs_parse_qconf.c
void dqs_parse_name_list(head,cp,opt)
dqs_list_type **head;
char *cp;
int opt;

1.129 dqs_parse_op(op_str)

File dqs_parse.c
int dqs_parse_op(op_str)
char *op_str;

1.130 dqs_parse_path_list(head,path_str)

File dqs_parse.c
void dqs_parse_path_list(head,path_str)
dqs_list_type **head;
char *path_str;
[host:]path[,[host:]path...]
str0 - path
str1 - host
str3 - temporary
int - temporary

1.131 dqs_parse_priority(priority_str)

File dqs_parse.c
int dqs_parse_priority(priority_str)
char *priority_str;

1.132 dqs_parse_qconf(argv_head,envp)

File dqs_parse_qconf.c

dqs_list_type *dqs_parse_qconf(argv_head,envp)

dqs_list_type *argv_head;

char **envp;

dqs_parse_qconf - is the parser utilized by ”qconf”

dqs_parse_qconf() returns a chained linked list of requests to

be forwarded, individually, to the qmaster.

1.133 dqs_parse_resource_list(head,resource_str)

File dqs_parse.c

void dqs_parse_resource_list(head,resource_str)

dqs_list_type **head;

char *resource_str;

ex: -l qty.eq.5,group=ibm,mem.gt.128,pvm

here's what we're gonna do:

we will refer to any ”-l blah=blah,blah2=blah2,...” as a ”resource_group”.

each ”resource_group” will consist of a ”chain” with the head being the qty.

by default ”qty=1”

op==.eq.——.ge.——.gt.——.lt.——.le.——.ne.

optional_field==numeric——string ¡a numeric of 0 is considered invalid¿

listel.int0=qty ¡default=1¿

listel.int1=chain_length

listel.chain=chained_element-¿

resource[.op.field][,resource[.op.field],...]

str0 - resource

str1 - [field]

int0 - [qty]

int1 - [op]

next-¿

resource[.op.field][,resource[.op.field],...]

str0 - resource

str1 - [field]

int0 - qty

int1 - [op]

next-¿

...

listel.next

:

:

NULL

1.134 dqs_parse_signal(signal_str)

File dqs_parse.c

int dqs_parse_signal(signal_str)

char *signal_str;

1.135 dqs_parse_states(state_str)

File dqs_parse.c

int dqs_parse_states(state_str)

char *state_str;

1.136 dqs_parse_user_list(head,user_str)

File dqs_parse.c

void dqs_parse_user_list(head,user_str)

dqs_list_type **head;

char *user_str;

str0 - user

str1 - cell

str3 - temporary

int3 - temporary

1.137 dqs_parse_variable_list(head,variable_str)

File dqs_parse.c

void dqs_parse_variable_list(head,variable_str)

dqs_list_type **head;

char *variable_str;

str0 - variable

str1 - value

str3 - temporary

int3 - temporary

1.138 dqs_parser_get_next(head,op_str)

File dqs_parse.c

```
dqs_list_type *dqs_parser_get_next(head,op_str)
dqs_list_type *head;
char *op_str;
```

1.139 dqs_print_group(gid)

File dqs_utility.c

```
void dqs_print_group(gid)
int gid;
```

1.140 dqs_q_master(sfd)

File dqs_sec.c

```
int dqs_q_master(sfd)
int sfd;
```

1.141 dqs_queue_available(q)

File dqs_select_queue.c

```
int dqs_queue_available(q)
dqs_queue_type *q;
```

1.142 dqs_read_in_conf_file()

File dqs_setup.c

```
void dqs_read_in_conf_file()
```

1.143 dqs_read_in_passwds(fname)

File dqs_reauth.c

```
dqs_list_type *dqs_read_in_passwds(fname)
char *fname;
```

1.144 dqs_read_in_qconf(file_prefix,file_name)

File dqs_queue.c

dqs_queue_type *dqs_read_in_qconf(file_prefix,file_name)
char *file_prefix;
char *file_name;

1.145 dqs_read_list_from_disk(file_prefix,file_name)

File dqs_list.c

dqs_list_type *dqs_read_list_from_disk(file_prefix,file_name)
char *file_prefix;
char *file_name;
dqs_read_list_from_disk - reads a dqs_list_type from disk.
follows chains as necessary.
returns bytes written
returns (-1) on error

1.146 dqs_read_resolve_file()

File dqs_resolve.c

void dqs_read_resolve_file()

1.147 dqs_read_rusage(f,

File dqs_job_exit.c

int dqs_read_rusage(f,
d, qcomplex_str)
FILE *f;
dqs_rusage_type *d;
char *qcomplex_str;
dqs_read_rusage - read rusage info from file.
Returns: -3 if f is NULL
-2 if the number of items read was incorrect
0 otherwise (i.e. success)
1 if EOF reached

1.148 dqs_readnbytes(sfd,

File dqs_io.c
int dqs_readnbytes(sfd,
ptr, n)
register int sfd;
register char *ptr;
register int n;
dqs_readnbytes - read n bytes from file descriptor

1.149 dqs_readstring(fd,ptr)

File dqs_io.c
int dqs_readstring(fd,ptr)
register int fd;
register char *ptr;
dqs_readstring - read string file descriptor

1.150 dqs_realloc(ptr,size)

File dqs_utility.c
char *dqs_realloc(ptr,size)
char *ptr;
int size;

1.151 dqs_reap()

File dqs_sig_handlers.c
void dqs_reap()

1.152 dqs_reap_children()

File dqs_sig_handlers.c
int dqs_reap_children()
several things have to be done here
** a) map child to jobnumber
** b) unlink spooled control file
** c) unlink spooled queue file
** d) gather statistics
** e) possibly send mail

** f) remove job from job list
** g) remove queue from queue list
** h) forward statistics back to qmaster

1.153 dqs_reauth(job)

File dqs_reauth.c
int dqs_reauth(job)
dqs_job_type *job;

1.154 dqs_recursive_rmdir(cp)

File dqs_tmpdir.c
void dqs_recursive_rmdir(cp)
char *cp;

1.155 dqs_remove_tmpdir(qconf,job)

File dqs_tmpdir.c
int dqs_remove_tmpdir(qconf,job)
dqs_queue_type *qconf;
dqs_job_type *job;

1.156 dqs_replicate_conf(config)

File dqs_replicate.c
dqs_conf_type *dqs_replicate_conf(config)
dqs_conf_type *config;

1.157 dqs_replicate_job(ojob)

File dqs_replicate.c
dqs_job_type *dqs_replicate_job(ojob)
dqs_job_type *ojob;

1.158 dqs_replicate_list(ohead,qty)

File dqs_replicate.c
dqs_list_type *dqs_replicate_list(ohead,qty)
dqs_list_type *ohead;
int qty;

1.159 dqs_replicate_me(mee)

File dqs_replicate.c
dqs_me_type *dqs_replicate_me(mee)
dqs_me_type *mee;

1.160 dqs_replicate_queue(queue)

File dqs_replicate.c
dqs_queue_type *dqs_replicate_queue(queue)
dqs_queue_type *queue;

1.161 dqs_replicate_rusage(rusage)

File dqs_replicate.c
dqs_rusage_type *dqs_replicate_rusage(rusage)
dqs_rusage_type *rusage;

1.162 dqs_resolve_cell(cell)

File dqs_resolve.c
dqs_list_type *dqs_resolve_cell(cell)
char *cell;

1.163 dqs_save_tid(lp,from_who)

File dqs_tid.c
void dqs_save_tid(lp,from_who)
dqs_list_type *lp;
int from_who;

1.164 dqs_schedule(job)

File dqs_schedule.c
int dqs_schedule(job)
dqs_job_type *job;

1.165 dqs_select_queue(req_lp,c_lp,q)

File dqs_select_queue.c
void dqs_select_queue(req_lp,c_lp,q)
dqs_select_queue - updates suitable queues based on requested resources vs specified complexes.
dqs_select_queue updates the "suitable" field of the queue if a hit is found in the request.

1.166 dqs_select_queues(req_lp)

File dqs_select_queue.c
void dqs_select_queues(req_lp)
dqs_select_queues - selects suitable queues based on requested resources vs specified complexes.
dqs_select_queues updates the "suitable" field of the Queue_list based on the number of "hits" found in the request_chain.

1.167 dqs_send_ack(sfd)

File dqs_send_receive.c
int dqs_send_ack(sfd)
int sfd;

1.168 dqs_send_list(cell,service,sfd,head)

File dqs_send_receive.c
int dqs_send_list(cell,service,sfd,head)
char *cell;
char *service;
int sfd;
dqs_list_type *head;
dqs_send_list_ - should only be called from dqs_send_list()
if (cell)

initiates connection
else
use fd

1.169 dqs_send_list_(host,service,sfd,head)

File dqs_send_receive.c
int dqs_send_list_(host,service,sfd,head)
```
char *host;
char *service;
int sfd;
dqs_list_type *head;
```
dqs_send_list_ - should only be called from dqs_send_list()
if (host)
initiates connection
else
uses sfd
returns sfd ¿= 0 on success (side effect of associating SFD)
-1 on failure (side effect of dissassociating SFD)

1.170 dqs_send_load()

File dqs_load_avg.c
int dqs_send_load()

1.171 dqs_send_mail(user,host,subj,buf)

File dqs_mail.c
void dqs_send_mail(user,host,subj,buf)
```
char *user;
char *host;
char *subj;
char *buf;
```

1.172 dqs_send_nak(sfd)

File dqs_send_receive.c
int dqs_send_nak(sfd)
```
int sfd;
```

1.173 dqs_send_receive_list(cell,service,out,in)

File dqs_send_receive.c
int dqs_send_receive_list(cell,service,out,in)
char *cell;
char *service;
dqs_list_type *out;
dqs_list_type **in;

1.174 dqs_set_coresize_2_0()

File dqs_reauth.c
void dqs_set_coresize_2_0()

1.175 dqs_set_coresize_back_normal()

File dqs_reauth.c
void dqs_set_coresize_back_normal()

1.176 dqs_set_defined_defaults()

File dqs_setup.c
void dqs_set_defined_defaults()

1.177 dqs_set_uid_gid(job)

File dqs_sec.c
void dqs_set_uid_gid(job)
dqs_job_type *job;

1.178 dqs_setenv(name,value,overwrite)

File dqs_setenv.c
int dqs_setenv(name,value,overwrite)
char *name;
char *value;
int overwrite;

1.179 dqs_setrlimits(qconf)

File dqs_exec_job.c

void dqs_setrlimits(qconf)

dqs_queue_type *qconf;

dqs_setrlimits - set process environment based on a queue configuration.

1.180 dqs_setup(dqs_formal_prog_name,qualified_prog_name)

File dqs_setup.c

void dqs_setup(dqs_formal_prog_name,qualified_prog_name)

u_long32 dqs_formal_prog_name;

char *qualified_prog_name;

1.181 dqs_setup_dqs_execd()

File dqs_setup.c

void dqs_setup_dqs_execd()

1.182 dqs_setup_qmaster()

File dqs_setup.c

void dqs_setup_qmaster()

1.183 dqs_setup_sig_handlers()

File dqs_sig_handlers.c

void dqs_setup_sig_handlers()

1.184 dqs_should_i_continue(ptr,head)

File dqs_parse_qconf.c

int dqs_should_i_continue(ptr,head)

char *ptr;

dqs_list_type *head;

1.185 dqs_show_acknak_list(head)

File dqs_list.c

void dqs_show_acknak_list(head)

dqs_list_type *head;

dqs_show_acknak_list - displays ACK/NAK messages

1.186 dqs_show_all_configurations()

File dqs_utility.c

void dqs_show_all_configurations()

1.187 dqs_show_checkpoint(how,op)

File dqs_utility.c

void dqs_show_checkpoint(how,op)

int how;

int op;

1.188 dqs_show_complex_list(head)

File dqs_utility.c

void dqs_show_complex_list(head)

dqs_list_type *head;

1.189 dqs_show_conf()

File dqs_setup.c

void dqs_show_conf()

1.190 dqs_show_hold_list(how,hold)

File dqs_utility.c

void dqs_show_hold_list(how,hold)

int how;

int hold;

1.191 dqs_show_job(job)

File dqs_parse.c
```
int dqs_show_job(job)
dqs_job_type *job;
```

1.192 dqs_show_keep_list(op,how)

File dqs_utility.c
```
void dqs_show_keep_list(op,how)
int op;
int how;
```

1.193 dqs_show_mail_options(op,how)

File dqs_utility.c
```
void dqs_show_mail_options(op,how)
int op;
int how;
```

1.194 dqs_show_me()

File dqs_setup.c
```
void dqs_show_me()
```

1.195 dqs_show_op(how,op)

File dqs_utility.c
```
void dqs_show_op(how,op)
int how;
int op;
```

1.196 dqs_show_qstat_request(job)

File dqs_show.c
```
int dqs_show_qstat_request(job)
dqs_job_type *job;
```

1.197 dqs_show_queue(q)

File dqs_queue.c

```
void dqs_show_queue(q)
dqs_queue_type *q;
```

1.198 dqs_show_queue_states(state)

File dqs_utility.c

```
int dqs_show_queue_states(state)
int state;
```

1.199 dqs_show_resource_list(head)

File dqs_utility.c

```
void dqs_show_resource_list(head)
dqs_list_type *head;
```

1.200 dqs_show_states(how,op)

File dqs_utility.c

```
void dqs_show_states(how,op)
int how;
int op;
```

1.201 dqs_show_y_n(op,how)

File dqs_utility.c

```
void dqs_show_y_n(op,how)
int op;
int how;
```

1.202 dqs_showlist(head,how,indent)

File dqs_list.c

```
void dqs_showlist(head,how,indent)
dqs_list_type *head;
int how;
int indent;
```

dqs_showlist - show linked list
head *dqs_list_type
how DQS_STDOUT—DQS_STDERR—DQS_STDERR—DQS_DEBUG
DQS_TYPE—DQS_INT0—DQS_INT1—DQS_INT2—DQS_INT3
DQS_STR0—DQS_STR1—DQS_STR2—DQS_STR3—DQS_CHAIN
indent how much to indent

1.203 dqs_shutdown()

File dqs_shutdown.c
void dqs_shutdown()

1.204 dqs_signal_queue(how,queue)

File dqs_c_qmod.c
int dqs_signal_queue(how,queue)
int how;
dqs_queue_type *queue;

1.205 dqs_sort_queues(head)

File dqs_list.c
dqs_list_type *dqs_sort_queues(head)
dqs_list_type *head;
dqs_sort_queues - sorts queues based on scheduling policy

1.206 dqs_strcasecmp(a,b)

File dqs_utility.c
int dqs_strcasecmp(a,b)
char *a;
char *b;
a number of libs don't include a strcasecmp() - hard to believe I know

1.207 dqs_string2list(str)

File dqs_list.c
dqs_list_type *dqs_string2list(str)
char *str;
dqs_string2list - converts a string to a list using whitespaces as IFS

1.208 dqs_string_insert(old_pointer,str)

File dqs_list.c

char *dqs_string_insert(old_pointer,str)

char *old_pointer;

char *str;

dqs_string_insert - insert a string on a pointer

mallocs necessary memory

if (old_pointer) free old_pointer

returns :

pointer to freshly malloced memory

1.209 dqs_strncasecmp(a,b,i)

File dqs_utility.c

int dqs_strncasecmp(a,b,i)

char *a;

char *b;

int i;

a number of libs don't include a strncasecmp() - hard to believe I know

1.210 dqs_strtok(str,delimitor)

File dqs_utility.c

char *dqs_strtok(str,delimitor)

char *str;

char *delimitor;

1.211 dqs_syslog(log_level,err_str)

File dqs_log.c

void dqs_syslog(log_level,err_str)

int log_level;

char *err_str;

1.212 dqs_tag_queues(job,qty,length)

File dqs_select_queue.c
int dqs_tag_queues(job,qty,length)
dqs_job_type *job;
int qty;
int length;
dqs_tag_queues - tags queues which are suitable for specified job should the "suitable" field of the queue be equal to the "length" of the requested resources, said queue is a possible match for the job. However, we must also verify the queue is available, has open slots and that the user is a vaild user of said queue.
dqs_tag_queues attempts to fill the request with as many different queues as possible.
returns number of slots filled.

1.213 dqs_terminate()

File dqs_sig_handlers.c
void dqs_terminate()

1.214 dqs_tid_del_x_host(host)

File dqs_tid.c
void dqs_tid_del_x_host(host)
char *host;

1.215 dqs_tid_garbage_collector()

File dqs_tid.c
void dqs_tid_garbage_collector()

1.216 dqs_trusted_host(sfd)

File dqs_sec.c
int dqs_trusted_host(sfd)
int sfd;

1.217 dqs_unlink(prefix,suffix)

File dqs_utility.c
void dqs_unlink(prefix,suffix)
char *prefix;
char *suffix;

1.218 dqs_unmap_signal(dqs_sig)

File dqs_utility.c
int dqs_unmap_signal(dqs_sig)
int dqs_sig;

1.219 dqs_unpack_conf(cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c
dqs_conf_type *dqs_unpack_conf(cur_ptr,bytes_unpacked,bytes_packed)
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;

1.220 dqs_unpack_header(buf_ptr,nibbles,total_bytes_packed)

File dqs_unpack.c
void dqs_unpack_header(buf_ptr,nibbles,total_bytes_packed)
char *buf_ptr;
int *nibbles;
int *total_bytes_packed;

1.221 dqs_unpack_job(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c
dqs_job_type *dqs_unpack_job(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)
char *buf_ptr;
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;

1.222 dqs_unpack_list(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c

dqs_list_type *dqs_unpack_list(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)

```
char *buf_ptr;
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;
```

!NOTE THAT mem_size may be greater than bytes_packed
THIS IS TO ACCOMONDATE DES!

1.223 dqs_unpack_me(cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c

dqs_me_type *dqs_unpack_me(cur_ptr,bytes_unpacked,bytes_packed)

```
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;
```

1.224 dqs_unpack_queue(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c

dqs_queue_type *dqs_unpack_queue(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)

```
char *buf_ptr;
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;
```

1.225 dqs_unpack_rusage(cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c

dqs_rusage_type *dqs_unpack_rusage(cur_ptr,bytes_unpacked,bytes_packed)

```
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;
```

1.226 dqs_unpack_token_list(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)

File dqs_unpack.c
dqs_list_type *dqs_unpack_token_list(buf_ptr,cur_ptr,bytes_unpacked,bytes_packed)
char *buf_ptr;
char **cur_ptr;
int *bytes_unpacked;
int bytes_packed;
!NOTE THAT mem_size may be greater than bytes_packed
THIS IS TO ACCOMONDATE DES!

1.227 dqs_update_qconf(descriptor,value)

File dqs_setup.c
void dqs_update_qconf(descriptor,value)
char *descriptor;
char *value;

1.228 dqs_update_queue_state()

File dqs_list.c
void dqs_update_queue_state()
dqs_update_queue_state - update the queues state based on lt_heard_from

1.229 dqs_usage()

File dqs_utility.c
void dqs_usage()

1.230 dqs_valid_queue_user(user,queue)

File dqs_sec.c
int dqs_valid_queue_user(user,queue)
dqs_valid_queue_user - determines if a user has access to a queue
if the queues exclude ACL list, xacl, is non-null, transverse
the list to see if the user is not explicitly denied access.
If the queues exclude ACL list, xacl, is non-null, transverse
the list to see if the user is not explicitly denied access.
returns 1 if user is a valid user;
0 if the user is not valid.

1.231 dqs_validate_option(opt,str)

File dqs_parse.c

void dqs_validate_option(opt,str)
int opt;
char *str;

1.232 dqs_wildmat(p,text)

File dqs_wildmat.c

int dqs_wildmat(p,text)
char *p;
char *text;
User-level routine. Returns TRUE or FALSE.

1.233 dqs_write_complex_template(ptr)

File dqs_parse_qconf.c

char *dqs_write_complex_template(ptr)
dqs_list_type *ptr;

1.234 dqs_write_conf_file(write_2_stdout,mconf)

File dqs_setup.c

void dqs_write_conf_file(write_2_stdout,mconf)
int write_2_stdout;
dqs_conf_type *mconf;

1.235 dqs_write_generic_qconf(file_prefix,file_name)

File dqs_queue.c

void dqs_write_generic_qconf(file_prefix,file_name)
char *file_prefix;
char *file_name;

1.236 dqs_write_job_to_disk(job)

File dqs_list.c

void dqs_write_job_to_disk(job)
dqs_job_type *job;

dqs_write_job_to_disk - writes dqs_job_type to disk

1.237 dqs_write_list_to_disk(file_prefix,file_name,list_head,qty)

File dqs_list.c

void dqs_write_list_to_disk(file_prefix,file_name,list_head,qty)
char *file_prefix;
char *file_name;
dqs_list_type *list_head;
int qty;
dqs_write_list_to_disk - writes up to "qty" elements to disk
"file_prefix" is optional
tpg

1.238 dqs_write_qconf(write_2_stdout,file_prefix,file_name,q)

File dqs_queue.c

char *dqs_write_qconf(write_2_stdout,file_prefix,file_name,q)
int write_2_stdout;
char *file_prefix;
char *file_name;
dqs_queue_type *q;

1.239 dqs_write_queue_to_disk(queue)

File dqs_list.c

void dqs_write_queue_to_disk(queue)
dqs_queue_type *queue;
dqs_write_queue_to_disk - writes dqs_queue_type to disk

1.240 dqs_write_rusage(f,

File dqs_job_exit.c

int dqs_write_rusage(f,
d, qcomplex_str)
FILE *f;
dqs_rusage_type *d;
char *qcomplex_str;
dqs_write_rusage - write rusage info to file.
Returns: -2 if f is NULL

number of characters written (¿ 0 indicates success)
EOF if failure

1.241 dqs_writenbytes

File dqs_io.c
int dqs_writenbytes
(sfd, ptr, n)
register int sfd;
register char *ptr;
register int n;
dqs_writenbytes - write n bytes to file descriptor

1.242 ex:

File dqs_parse_qconf.c
/* ex:
mem=5
group=ibm
arc=RIOS
pvm
here's what we're gonna do:
each "resource_group" will consist of a "chain" with the head-¿str0
being the "complex_name".
listel.str0="complex_name"
listel.int1=chain_length
listel.chain=chained_element-¿
resource[=field][,resource[=field],...]
str0 - resource
str1 - [field]
int0 - [qty]
int1 - [op]
next-¿
resource[=field][,resource[=field],...]
str0 - resource
str1 - [field]
int0 - qty
int1 - [op]
next-¿
...
listel.next

:
:
NULL
dqs_list_type *dqs_read_complex_file(prefix,fname)
char *prefix;
char *fname;

1.243 int

File dqs_utility.c
unsigned int
dqs_rand(min,max)
int min;
int max;

1.244 int

File dqs_wildmat.c
static int
dqs_DoMatch(text, p)
register char *text;
register char *p;
Match text and p, return TRUE, FALSE, or ABORT.

1.245 int

File dqs_wildmat.c
static int
dqs_Star(text, p)
register char *text;
register char *p;
See if the text matches the p, which has an implied leading asterisk.