**These should eventually become git issues**

* we will do the same (i.e. have distinct Makefile targets) if we do a db2 version of this for HPSS, it will need to be a separate makefile macro DB2
* add summarize the directory by user and add user records to bfwi  
  + this has been done more or less but not to bfwi - it is a query in the scripts dir that walks and runs insert from select queries to create by user summary should consider making a callable utility for this so it could be done from bfwi/bfmi/etc. the script/query version might be fast enough that this is not needed, need to try to run it at scale
* add summarize the directory summaries by user and add records to bfti  
  + this has been done more or less but not to bfwi - it is a query in the scripts dir that walks and runs insert from select queries to create by group summary should consider making a callable utility for this so it could be done from bfwi/bfmi/etc. the script/query version might be fast enough that this is not needed, need to try to run it at scale
* add "by user" and "by group" summaries to tree summary in bfti
* put on LANL headers on all .c's and ack the threadpool package  
    
  [Proposed text is in COPYRIGHT.txt. But should it say "this file is part of GUFI ...", or "This file is part of MarFS ..."?]
* clean up error checking and messages

  + etc?
* need a large functional testsuite (this is started in the test dir but we need a create tree, bfwi, bfti, bfq, querydb, querydbn, uid and gid summary and query, oldbigfiles )
* consider building a DSL for stringing sql statements/output/result/and.or together to be run at init/per thread, per thread/dir, and fin/per thread
* try new persistent version of bfli reading multiple files?
* add whatever needed to bfmi
* try an hpss load with persistent version bfdi/bfhi that reads from db2 or from db2 dumps
* Some form of feedback to the user, about ongoing progress.
  + assure that problems that imply failure to do the job correctly will result in abort.
* ability to restart
* reduce struct copying. Lots of places (e.g. pushn() in structq.c) where structs are copied into other structs. Seems like it should be simple to either store the pointer into the destination struct, or just use the source struct directly and eliminate the need for a distinct destination struct.
* reduce sprintfs(), where strcpy() would work
* Currently, bfwi source=/a/b/c, dest=/this/is/the/dest, will build the GUFI tree starting at /this/is/the/dest/a/b/c. This latter is then what must be provided to bfti, bfq, etc. If bfwi built instead directly in /this/is/the/dest, then that same arg could be passed to bfti/bfq. (You could still get the current behavior by providing dest=/this/is/the/dest/a/b/c.  
    
  The way to do this would be to capture the length of the source-dir in bfwi's processinit(), and store that in the work struct. Then processdir() -> dupdir() would append name+length onto in.nameto.
* Use a bloom filter instead of min/max uid/gid in summary/treesummary schemas. This might allow more-accurate elimination of unnecessary sub-trees, when searching the tree for particular users and/or groups.