00readme

The RESA program, as of December 2018, was last configured to run on Linux 6.9 on an x86 processor. This document captures some of the important lessons learned in order to support RESA.

Currently RESA is compiled on the machine which it runs. This is in lieu of copying the executables to the target platform (by the way, executables are identified with the extension **.run**). Which means each server is compiled separately. AFTER each compile be sure that access to the executables is to the group, or workstations will not have access. E.g. chmod 750 \*.run. The workstations ssh to the resa server and execute locally on the server.

RESA is written in RATFOR( an extension of FORTRAN) and C. The RATFOR code is converted into Fortran, which is then compiled before being linked into an executable. E.g. ratc \*.rat then forc \*.for followed by the link (lnk) commands.

RESA most recently was ported from a DEC ALPHA platform running VMS operating system to Linux around 2013-2014. There will be some comments within code and the attached documents that are obsolete or inapplicable (Staff for the last port was minimal, so not everything was kept up-to-date). RESA communicates to the JLCCTC federation through an adaptor (MITRE product) which effectively translates the RESA (ALSP) syntax into HLA.

RESA uses a shared area of data. In previous operating systems this may have been called a globalsec, for resa it is called the blackboard. All resa, with the exception of terrain, is stored within the blackboard in a “bit packed” fashion i.e. each word of data may have multiple pieces of information. Multiple executables share the same source files to allow a common “knowledge” of the blackboard and the “how to” for retrieving and storing information.

* NWISS\_DB is the blackboard for the RESA game. If trying to access this when one doesn’t exist (e.g. in wrong directory) a file will be created; which can be deleted. The file may be simply copied to create backups, save point, or to port to another machine.
* DTED.AA is the terrain file. This file is **dynamically** changed, so any changes made via orders will carry forward to next usage. i.e. Make a backup before manipulating the data with orders. Also “ground” is essentially any area defined with >0 elevation.
* Any changes to the size or number of items make old saves invalid as the NWISS\_DB mapping changes. There are several existing size limitations as well, so care should be taken when making modifications. A change was made in bbrte.def [if looking at a basecode directory, notice that many items are compiled/created as part of compilation.]

Installation.

* Install either CENTOS or REDHAT. The included attachment, ***Resa installation.docx****,* will walk through basic installation/unpacking of the RESA software and needed compilers. Development support has been on CENTOS with users running beneath REDHAT.
* There are TWO code tarballs included. The first is supportive of user environments and does NOT include a second branch of data.
  1. resa\_build – branch of code and data for user installation. This is addressed in ***Resa installation*.docx**
  2. resa\_cm – branch of code to support a more rigorous cm solution. A full build from this branch includes more components/sub-component/library compiled.
* There are TWO intel compilers available. The one in directory ***2016\_intel\_compiler*** is the one used through Dec 2018, the other in ***2018\_intel\_compiler*** has not been operationally used. Neither of these have a current support agreement/contract with Intel.
* The second tree, resa\_cm, is a configuration management structure where a full and complete build can be made (i.e. the user side is completely deleted and repopulated with every possible item rebuilt). DO NOT do a full CM build without first verifying the current user tree source code is present beneath the developer tree. If not verified, the current “in use” code will be lost. If installing the developer resa tarball, some adjustments may be needed to accommodate additional directory layers.
* The local resa account **.bashrc** sources a provided **resabashrc** file, which provides all the aliases and other defines of importance.

Execution.

* See the included documents ***RESA\_tech control.docx*** and ***running\_resa\_in\_gdb.txt***

Compilation.

* MAKE\_TRUNK command will create all of the .RUN executables. This can be used to build everything during an exercise. In theory can compile and link each file/component separately, but it sometimes didn’t work consistently, especially when operating within gdb (debugger). GDB\_DOCUMENT??
* gresa is a short cut to the basecode
* gwar is a short cut to the game directory
* After compiling code, chmod 750 the \*.run files in the code directory (gresa). Otherwise remote workstations may not be able to access executables.
* In the top level directory of resa\_cm is a script ./top\_build\_script which will empty the resa\_build directory, copy files into it from resa\_cm, then compile and link components.

Code

* The majority of the source code can be found in $RESA\_TRUNK directory (alias gresa will go there). Currently ***resa\_build/resang/trunk/*** has a rudimentary ***00all.log*** file exists to indicate changes made and why, during 2016-2018. Also available are previous versions of files. Any KBSC old disks may have other baselines (often stored as zip files), the version included here is streamlined to only one version.
* Code changes will usually be to a .rat, or .c file. A few scattered files are in other subdirectories, but this is the where nearly all changes should occur (riskier when made elsewhere).
* Because the blackboard is of fixed and allocated size, there are limits to how many of different things which may be stored/accessed/used.
* Objects are defined by bb\*def files. The other bb\*files are created as part of the build process. E.g. ***bbunt.def, bbunt.inc, bbunt.tst***. The Xtract and Put (store) command for each variable is defined in the .inc version of the file. Changes to bb\*def files will invalidate saves.
* RESA was maintained in a shared github.com account to allow corroboration development. It MAY be possible to gain access by request in the future.

Debugging.

* BBTEST is a utility that allows access to the blackboard where data may be inspected and even modified if desired. Instructions are within attached file Consolidated docs.docx (heading: ***resa\_programming.doc*** ). Use the bb\*def files to determine WHERE to look for data and under what variable name. In addition part of the source file for this has been cut out and made available as a text document, ***btmain.txt***.
* The executables for resa may be run within GDB where local values may be inspected and modified. In addition, a crash traceback may be generated if desired/needed using gdb commands.
* swar (the war simulator).
  + gdb wgmain.run
  + {set any desired breakpoints, see gdb help on line}
  + Run
* Workstation
  + gdb wsmain.run
  + set args 1 t as=1 config=cciox
  + {set any desired breakpoints}
  + run

Air Tasking Orders (ATO)

* The ability for RESA to process Air Tasking Orders (ATO) as sent from AWSIM was a desired capability that was temporarily lost. The attached document ***ATO Procedure 1.docx*** details how to process these messages. A file of routes and air missions is sent, which can be processed and executed by RESA therefore removing the need for manual entry by augmentees.

Compliance set up/install modifications

* The attached document, ***resa logins.docx***, shows a methodology necessary to implement unique individual controller logins and to track (show) login history. The document also shows how to minimize the access to various files and directories (i.e. remove all world access to files and directories; and to minimize group access).
* A second document, ***Install antivirus.docx***, shows how to modify an installation script to allow for RESA’s anti-virus to be updated from the STRICOM provided Linux support disk. Currently the initial installation is part of the initial installation and lockdown steps.
* The lack of an Authorization To Operate (a second ATO acronym) was final justification to end use. To regain accreditation, interface with STRICOM and focus on using login authorization software. The focus of accreditation is that the system when installed controls user access to only information/places authorized. Not as concerned with does user do any harm to the application. i.e. Who accessed files, was logged in, is concern. Who issued a command that caused simulation to crash, not as important. May be part of same solution, but 2nd issue isn’t an accreditation issue. Previous to September 2018, RESA ran beneath STRICOM installation and lockdown configured to support “WARSIM/WIM System WITH Oracle client SW". In this configuration, no separate effort to install RPMs was needed and GDB worked as expected. Although not all of the known issues (non-compliancy) was shared with the developer a few known/suspected items are known. Following are suspected issued that would need addressed to revive RESA:
  1. Password complexity. The OS as delivered, PEOSTRI lockdown, has appropriate rules for length and complexity.
  2. File access. Preference is that 750 is the most liberal. Currently some 770 protections are needed (see documentation)
  3. Individual logins and tracking.
* The attached folder, ***RESA\_SECURITY,*** has two files in it that were provided September 2018 by a MITRE representative. These were the first hints as to what RESA would need to comply with in order to receive an ATO. The understanding is that priority 3 items are NOT required per DOD. For other issues that could not be complied with, a waiver would have been needed. In past cases, the waiver was based upon cost/benefit/risk and/or feasibility arguments. Needless to say, other than as documented above there was not a tasked effort for RESA to comply.

Other documents on the DVD

* Resa guide – a directory containing multiple sections of the resa guide. This was NOT completely kept up to date for the past several years (maybe 2013 or before). One notable section involved ATO (air tasking order) processing. This is addressed by a separate document.
* ***SPAWAR\_GUIDE.docx*** - a version of the controller training document SPAWAR had funded in the past.
* ***Consolidated text documents.docx*** – This contains several text documents that had been prepared by previous developers. These contain a wide range of topics.
* ***Consolidated docs.docx*** – This contains several documents that had been prepared by previous developers. These cover a variety of topics and have had a few sections called out in the above text.

KNOWN ISSUES

* Never conquered the ability to modify enumerations files. This was a high priority for resolving if RESA continued its usage. Note: final two years of support was 1 part time programmer. Any notes found that imply a capability to automatically update is incorrect, the changes must be manual with an unknown follow up to generate items to be compiled. Unverified procedure is:
  1. Hand edit ***enum-jlcctc.h*** and add/subtract changes
  2. Hand edit **alsp\_*enumeration\_data.rin***
  3. Issue command **enumnum**
  4. Compile all code. i.e. **make\_trunk**
* At last use, the GDB debugger was not reliably able to provide traceback information (failed 2 cases overnight, but worked in all subsequent cntrl-c with bt tests). A quick test, bring up executable in GDB. After issuing **run**, do a **cntrl-c**, then request **BT** (backtrace). Suspicion is that RESA may have gotten into a recursive calling loop, but not verified.
* Most other known issues were nice-to-haves. Better GUI. Ability to select graphic symbol colors. See attached document, ***resa issues.docx***, for some of the remaining open items.

Good to know

* Variable names are not case sensitive in ratfor/fortran
* The use of $ in a variable name is just a character (nothing special)
* The cross-model linkage had some logic/realism errors. I.e. torpedo engagements didn’t support detection and effective counter-measures, unless this was implicitly enacted with PKs and logic.
* RESA has a capability to “ghost/replicate” ROK sensors. This allowed RESA to provide a RESA (US) and CH(Korea) view through GCCS to COP (common operation picture). ROK do not want to write code to feed a US COP product. Joint commander wanted (wants) that view.
* There are several utilities, similar to bbtest, that create reports or do other things. Not all have short cut defined in the resabashrc file four of these are:
  1. $RESA\_ROOT/pasmain.run
  2. $RESA\_ROOT/pfmain.run
  3. $RESA\_ROOT/pwmain.run
  4. $RESA\_ROOT/pomain.run
* The date set in the .ini file is Julian date, however the order SET ZULU can be used to move hour and minutes forward/backward to match real zulu.
* Unfortunately, I don’t possess the information regarding database setup, configuration, etc. Mr. Kon Shin is the last person to know these processes. There is documentation in the RESA Guide, but it may not be completely current.

tmb