art - Support #19287

un-demangleable symbol

03/06/2018 12:24 PM - Christopher Backhouse

Status:	Closed	**Start date:**	03/06/2018
Priority:	Normal	**Due date:**	
Assignee:	Kyle Knoepfel	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Scope:	Internal	**SSI Package:**	
Experiment:	-		

Description

Debugging with gdb I got this:

```
/home/greenc/work/cet-is/test-products/gdb/v7_12/src/gdb-7.12/gdb/cp-support.c:1615: demangler-warning: unable to demangle '_ZSt7forwardIRZN3art19TriggerNamesServiceC4ERKN5fhicl12ParameterSetERKSt6vectorINSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEESaISC_EEEUlRT_mRKT0_E_EOSH_RNSt16remove_referenceISH_E4typeE' (demangler failed with signal 11)
A problem internal to GDB has been detected,
further debugging may prove unreliable.
Quit this debugging session? (y or n) n
```

things seemed to work OK after I continued, but it's an inconvenience.

SL6's built in c++filt can't understand this symbol either, but it doesn't crash.

Related issues:

Is duplicate of art - Support #17751: a problem with debugging art v2_07-base...	**Closed**	**09/20/2017**

History

#1 - 03/07/2018 08:16 AM - Kyle Knoepfel

- Tracker changed from Bug to Support

- Status changed from New to Feedback

gdb 7.12 is known to suffer from the inability to demangle symbols generated by more modern compilers. Please use gdb 8.0.1 instead and let us know if there is still an issue (http://scisoft.fnal.gov/scisoft/packages/gdb/v8_0_1/).

#2 - 03/07/2018 08:17 AM - Kyle Knoepfel

- Has duplicate Support #17751: a problem with debugging art v2_07-based executables with gdb added

#3 - 03/07/2018 08:17 AM - Kyle Knoepfel

- Has duplicate deleted (Support #17751: a problem with debugging art v2_07-based executables with gdb)

#4 - 03/07/2018 08:18 AM - Kyle Knoepfel

- Is duplicate of Support #17751: a problem with debugging art v2_07-based executables with gdb added

#5 - 03/07/2018 08:21 AM - Christopher Backhouse

Seems like this is just an oversight on our side. We explicitly specify gdb v7_12.

Is there some source for a newer c++filt we could use, or a replacement command?

#6 - 03/07/2018 08:29 AM - Kyle Knoepfel

Chris Green has provided an SLF6 version for DUNE. I've copied it here:

novagpvm01.fnal.gov:~knoepfel/c++filt

We have not yet found a way to generally package it, though.

#7 - 03/07/2018 08:35 AM - Christopher Backhouse

Works, though I'm not sure it's a big improvement :)

Thanks!

```
$ ~knoepfel/c++filt _ZSt7forwardIRZN3art19TriggerNamesServiceC4ERKN5fhicl12ParameterSetERKSt6vectorINSt7__cxx1
112basic_stringIcSt11char_traitsIcESaIcEEESaISC_EEEUlRT_mRKT0_E_EOSH_RNSt16remove_referenceISH_E4typeE

art::TriggerNamesService::TriggerNamesService(fhicl::ParameterSet const&, std::vector<std::__cxx11::basic_stri
ng<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<char, std::
char_traits<char>, std::allocator<char> > > > const&)::{lambda(auto:1&, unsigned long, auto:2 const&)#1}& std:
:forward<art::TriggerNamesService::TriggerNamesService(fhicl::ParameterSet const&, std::vector<std::__cxx11::b
asic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx11::basic_string<ch
ar, std::char_traits<char>, std::allocator<char> > > > const&)::{lambda(auto:1&, unsigned long, auto:2 const&)
#1}&>(std::remove_reference<art::TriggerNamesService::TriggerNamesService(fhicl::ParameterSet const&, std::vec
tor<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, std::allocator<std::__cxx
11::basic_string<char, std::char_traits<char>, std::allocator<char> > > > const&)::{lambda(auto:1&, unsigned l
ong, auto:2 const&)#1}&>::type&)
```

#8 - 03/07/2018 08:38 AM - Kyle Knoepfel

- Status changed from Feedback to Closed

Agreed. :)

If you think it's worth having a packaged c++filt, then please go ahead and fill out a feature request.

#9 - 03/07/2018 08:38 AM - Kyle Knoepfel

- Assignee set to Kyle Knoepfel

- % Done changed from 0 to 100