Build and install ICU on IBM i

A step-by-step introduction to building ICU4C on IBM i

Zhang Jin Jiang
Li Jia Chen

November 20, 2012

International Components for Unicode (ICU) plays a key role in developing a globalized product. ICU is supported by option 39 of the base IBM i, however, you may want to build your customized version of ICU and include it together with your product. Building ICU on IBM i is as easy as building it on other platforms, but you may still experience some problems due to the platform difference. To avoid those problems, some environment variables should be set correctly. In this article, we introduce the steps on how to build ICU on IBM i and how to fix the potential problems when building it.

Overview of ICU

ICU is a mature, portable set of C/C++ and Java™ libraries providing Unicode and globalization support for software applications across many platforms. As an open source project sponsored by IBM, it offers great flexibility for developers to use and customize the provided services, including: code page conversion, collation, formatting, time calculations and so on.

ICU supports a variety of platforms, including Microsoft® Windows®, Linux®, IBM AIX®, Solaris, Mac OS X, IBM i, and so on. As there are a number of versions of compilers on so many platforms and the compatibility between them could not be guaranteed, only a limited number of binary versions of ICU are distributed. For the platforms on which there are no binary versions provided, we could build ICU from the source and customize the ICU libraries to meet our needs.

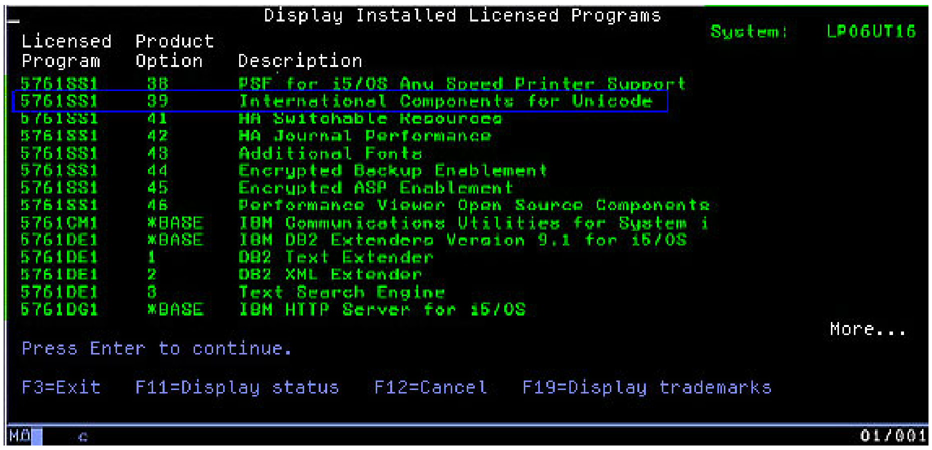
Although there are both C/C++ libraries (ICU4C) and Java libraries (ICU4J) available in the ICU project, building ICU4J would not be covered in this article, and we will refer to ICU4C by saying ICU in this article.

ICU support for IBM i

ICU is provided as one option in the base part of the IBM i operating system. Considering IBM i 6.1 as example, the Licensed Program ID of ICU is 5761SS1 and the Product Option is 39. To check the installation status, use the GO LICPGM command and in the Work with Licensed Programs menu, select option **10**.

Press the F11 key twice to display the product option.

Figure 1. ICU installation status



```
                        Display Installed Licensed Programs
                                                            System:   LP06UT16
 Licensed  Product
 Program   Option    Description
 5761SS1    38       PSF for i5/OS Any Speed Printer Support
 5761SS1    39       International Components for Unicode
 5761SS1    41       HA Switchable Resources
 5761SS1    42       HA Journal Performance
 5761SS1    43       Additional Fonts
 5761SS1    44       Encrypted Backup Enablement
 5761SS1    45       Encrypted ASP Enablement
 5761SS1    46       Performance Viewer Open Source Components
 5761CM1    *BASE    IBM Communications Utilities for System i
 5761DE1    *BASE    IBM DB2 Extenders Version 9.1 for i5/OS
 5761DE1    1        DB2 Text Extender
 5761DE1    2        DB2 XML Extender
 5761DE1    3        Text Search Engine
 5761DG1    *BASE    IBM HTTP Server for i5/OS
                                                                      More...
 Press Enter to continue.

 F3=Exit   F11=Display status   F12=Cancel   F19=Display trademarks

MA    c                                                                01/001
```

ICU is one option of the base part of the IBM i operating system, but because ICU is ported to the IBM i platform and rarely is provided as a binary distribution, it is necessary to build ICU yourselves if you need to customize ICU libraries and include it together with your product. For example, if you are using ICU in a memory-constrained environment, and you need only the Unicode-based collation library that is provided by ICU, then you can remove the other data libraries to bring down the size of the whole ICU package to a fraction of the original size.

Building ICU on IBM i is as easy as building it on other platforms, however, you may still experience some problems due to the platform difference. To avoid those problems, some environment variables should be set correctly. In this article, we introduce detailed steps on how to build ICU on IBM i as well as how to fix the potential problems when building it.

Note that we use ICU 4.4.2 and IBM i 6.1 for this article. Although not all combinations of ICU version level and IBM i operating system level were evaluated while writing this article, we believe that most of the issues you might encounter will be similar to the topics covered throughout this article.

Setting up the building environment

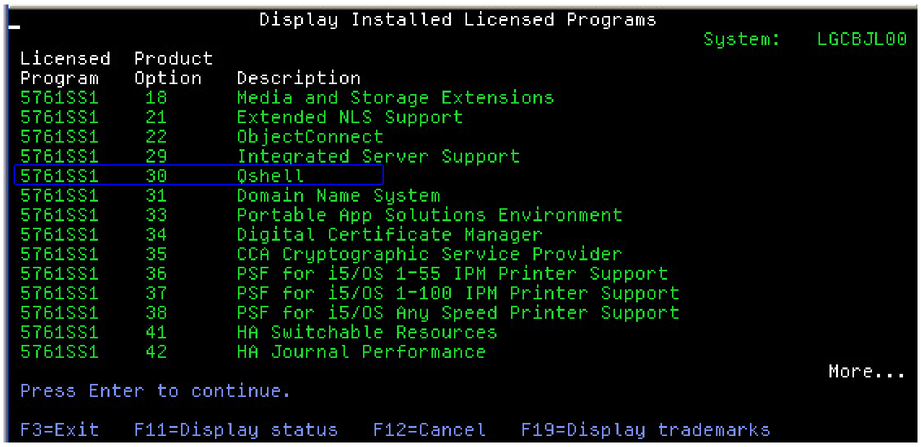
Before we start building ICU on the IBM i platform, the following requirements need to be met.

Install Qshell interpreter

Qshell is an option of the base part of the IBM i platform as well, which is Licensed Program 5761SS1 and Product Option 30 on IBM i 6.1. It is a command-line environment that is built on IBM i, providing an interface similar to UNIX®, so that we can run the shell script on IBM i for building ICU.

We can use similar commands to view the installation status of Qshell as what we did for ICU earlier.

Figure 2. Qshell installation status

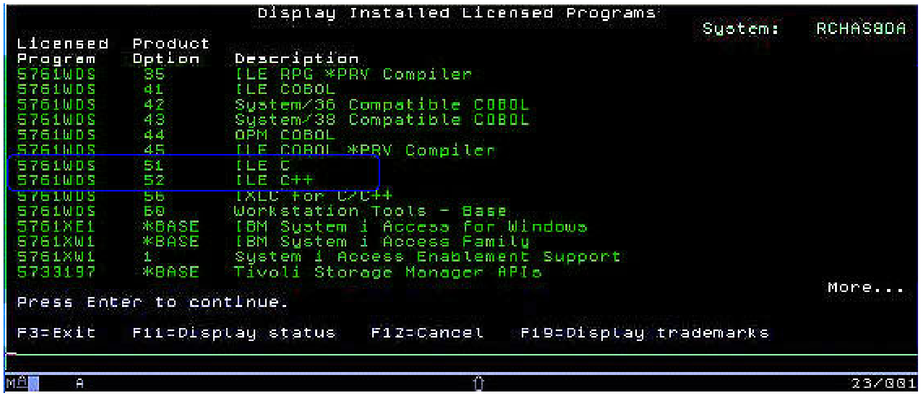


```
                      Display Installed Licensed Programs
                                                            System:   LGCBJL00
 Licensed  Product
 Program   Option    Description
 5761SS1    18       Media and Storage Extensions
 5761SS1    21       Extended NLS Support
 5761SS1    22       ObjectConnect
 5761SS1    29       Integrated Server Support
 5761SS1    30       Qshell
 5761SS1    31       Domain Name System
 5761SS1    33       Portable App Solutions Environment
 5761SS1    34       Digital Certificate Manager
 5761SS1    35       CCA Cryptographic Service Provider
 5761SS1    36       PSF for i5/OS 1-55 IPM Printer Support
 5761SS1    37       PSF for i5/OS 1-100 IPM Printer Support
 5761SS1    38       PSF for i5/OS Any Speed Printer Support
 5761SS1    41       HA Switchable Resources
 5761SS1    42       HA Journal Performance
                                                                       More...
 Press Enter to continue.

 F3=Exit   F11=Display status   F12=Cancel   F19=Display trademarks
```

Install C/C++ compiler

The Integrated Language Environment (ILE) C/C++ compilers, which are parts of the WebSphere Development Studio for i, need to be installed. They are Licensed Program 5761WDS, Product Option 51 and 52 on IBM i 6.1. ILE C and C++ compilers support the development on IBM i in both C and C++ programming languages, which is important for building ICU.

Figure 3. ILE C/C++ compiler installation status



```
                      Display Installed Licensed Programs
                                                            System:   RCHAS8DA
 Licensed  Product
 Program   Option    Description
 5761WDS    35       ILE RPG *PRV Compiler
 5761WDS    41       ILE COBOL
 5761WDS    42       System/36 Compatible COBOL
 5761WDS    43       System/38 Compatible COBOL
 5761WDS    44       OPM COBOL
 5761WDS    45       ILE COBOL *PRV Compiler
 5761WDS    51       ILE C
 5761WDS    52       ILE C++
 5761WDS    56       IXLC for C/C++
 5761WDS    60       Workstation Tools - Base
 5761XE1    *BASE    IBM System i Access for Windows
 5761XW1    *BASE    IBM System i Access Family
 5761XW1    1        System i Access Enablement Support
 5733197    *BASE    Tivoli Storage Manager APIs
                                                                       More...
 Press Enter to continue.

 F3=Exit   F11=Display status   F12=Cancel   F19=Display trademarks

 MA      A                                                             23/001
```

Install and set up IBM Tools for Developers for IBM i

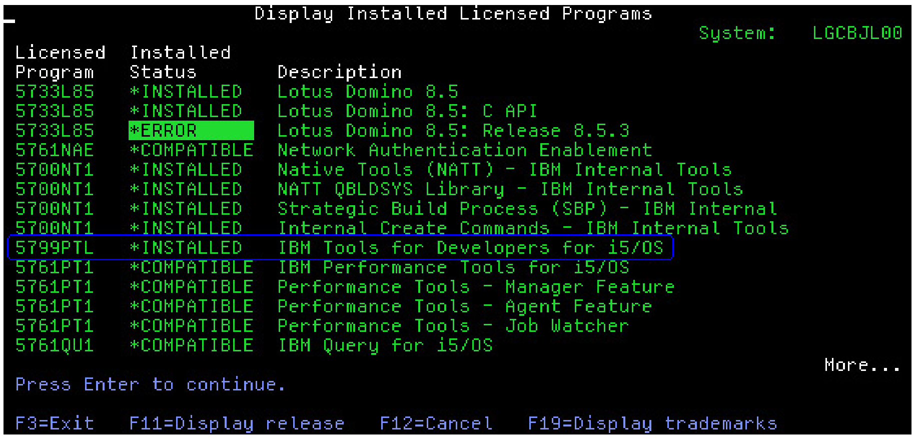
The IBM Tools for Developers for IBM i needs to be installed, which is a no-charge programming request for price quotation (PRPQ) product that contains various tools. Many of these tools are ported from other platforms as they are popular and useful for developers. These tools play critical roles by aiding in development, building, porting, and deployment of the IBM i applications and improving developer's productivity, because they take advantage of the different environments on IBM i, such as the command line, Qshell, and the Portable Application Solutions Environment for i (PASE for i), and make developers being able to remain in a certain environment to work with the provided tools.

Some of examples of these tools include: icc (a complier that invokes ILE C or ILE C++ compiler from Qshell), gmake (a GNU version of make), GNU emacs (an extensible text editor), gzip (a popular data compression program of GNU), qar (a utility for creating, modifying, and extracting from archives), GNU gawk (a pattern-matching utility) and so on.

To install IBM Tools for Developers for IBM i, we can download it from the IBM Tools for Developers for IBM i website and perform the following steps:

1. Extract the file to get q5799ptl_v5r4m0.savf
2. Create a Save File on IBM i using the following command.
   ```
   CRTSAVF FILE (QGPL/Q5799PTL)
   ```
3. Upload the file to IBM i using FTP, and FTP commands such as:
   ```
    put q5799ptl_v5r4m0.savf QGPL/q5799ptl
   ```
4. Install IBM Tools for Developers for IBM i on IBM i using the following commands:
   ```
    RSTLICPGM LICPGM(5799PTL) DEV(*SAVF) LNG(2924) SAVF(QGPL/Q5799PTL)
   ```
5. View the installation status using the following command on IBM i, and use option 10 and then press F11 twice to display the product option.
   ```
   GO LICPGM
   ```

Figure 4. IBM Tools for Developers for IBM i installation status



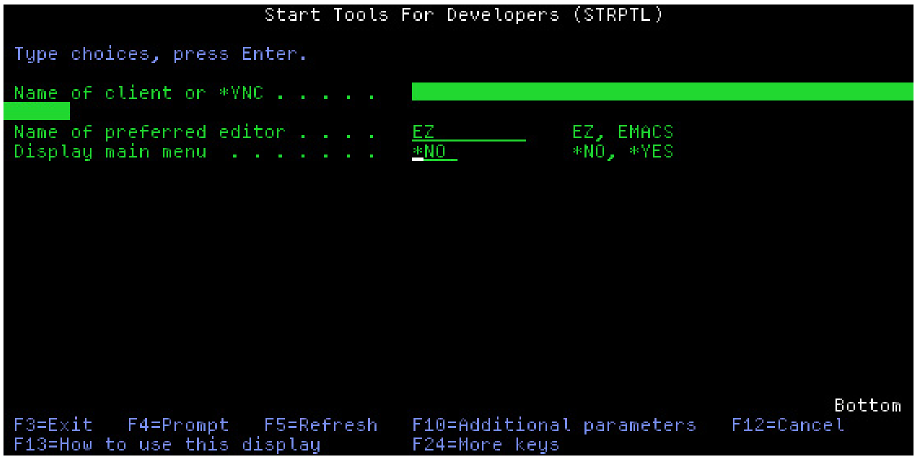
```
                      Display Installed Licensed Programs
                                                              System:   LGCBJL00
Licensed  Installed
Program   Status        Description
5733L85   *INSTALLED    Lotus Domino 8.5
5733L85   *INSTALLED    Lotus Domino 8.5: C API
5733L85   *ERROR        Lotus Domino 8.5: Release 8.5.3
5761NAE   *COMPATIBLE   Network Authentication Enablement
5700NT1   *INSTALLED    Native Tools (NATT) - IBM Internal Tools
5700NT1   *INSTALLED    NATT QBLDSYS Library - IBM Internal Tools
5700NT1   *INSTALLED    Strategic Build Process (SBP) - IBM Internal
5700NT1   *INSTALLED    Internal Create Commands - IBM Internal Tools
5799PTL   *INSTALLED    IBM Tools for Developers for i5/OS
5761PT1   *COMPATIBLE   IBM Performance Tools for i5/OS
5761PT1   *COMPATIBLE   Performance Tools - Manager Feature
5761PT1   *COMPATIBLE   Performance Tools - Agent Feature
5761PT1   *COMPATIBLE   Performance Tools - Job Watcher
5761QU1   *COMPATIBLE   IBM Query for i5/OS
                                                                     More...
Press Enter to continue.

F3=Exit   F11=Display release   F12=Cancel   F19=Display trademarks
```

After installing the tools successfully, we can use the following command to start the tools. The command initializes certain environment variables and sets up related tasks for the user to use this tool.

```
STRPTL
```

When the following screen is displayed, type the name of the client. This name can be used to ensure that the X-window screens are sent to the correct display of the client.

Figure 5. Start IBM Tools for Developers for IBM i



```
                       Start Tools For Developers (STRPTL)

Type choices, press Enter.

Name of client or *VNC . . . . .

Name of preferred editor . . . .   EZ            EZ, EMACS
Display main menu  . . . . . . .   *NO           *NO, *YES










                                                                         Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Upload ICU source code files to IBM i IFS

Then we can upload the ICU source code files to IBM i using the following commands. (In this case, consider ICU4.4.2 as an example.) Note that in order to unpack ICU and convert the files to an EBCDIC code page to restore the binary files can be restored to the original code page, we need to copy unpax-icu.sh from the ICU source ZIP file (under folder ..\icu\as_is\os400) and upload it to IBM i as well.

```
ftp [serverip]
[Username]
[Password]
quote site namefmt 1
binary
put icu4c-4_4_2-src.tgz /tmp/zjj/icutemp/icu4c-4_4_2-src.tgz
put unpax-icu.sh /tmp/zjj/icutemp/unpax-icu.sh
```

Changing the binary mode in the command is important as it avoids problems that might be caused by different code pages between IBM i and the platforms that you downloaded ICU to.

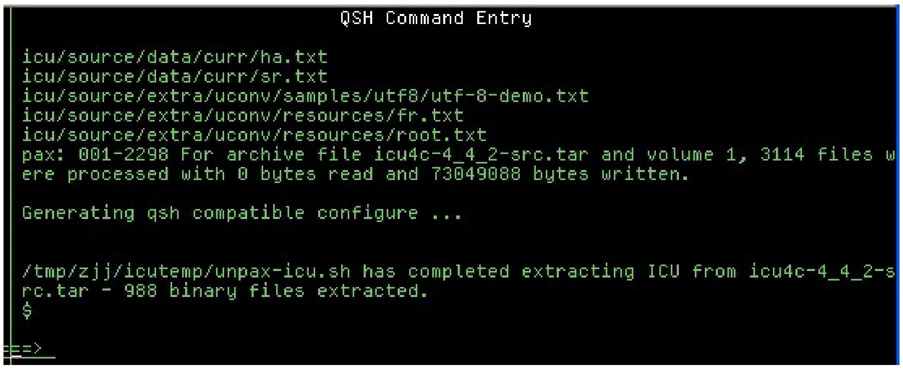
Building ICU on IBM i

After the building environment is ready, we can start to build ICU on IBM i. The steps here are a little different from the ones introduced in the ICU *readme* file. Actually, that is one of the points that this article will cover. To build ICU4C 4.4.2 successfully on IBM i, we should:

1. Create a library where you want to store the building results, that is, ICU utility programs, service programs, and modules. In thiscase, I want to put the results in iculib6, so you create the following library:
   ```
   CRTLIB LIB(iculib6)
   ```
2. Add the necessary environment variables that are used by ICU to configure and make process. A series of variables are needed, and they are:
   ```
   addenvvar envvar(OUTPUTDIR) value('iculib6') replace(*YES)
   addenvvar envvar(MAKE) value('gmake') replace(*YES)
   addenvvar envvar(GREP) value('grep') replace(*YES)
   addenvvar envvar(AR) value('qar') replace(*YES)
   addenvvar envvar(AWK) value('gawk') replace(*YES)
   ```
3. After adding the variables, you can run the `wrkenvvar` command to check your results.

4. Change this job's CCSID to 37 by using the following command:
```
chgjob ccsid(37)
```
5. Start Tools For Developers using the `strptl` command. This tool provides some of the useful utilities that are needed for building ICU and including icc, gmake, qar, and so on.
6. Start the shell by running the `Qsh` command and change to the directory where the ICU source code is located. In this case, it is in the /tmp/zjj/icutemp/ directory. Further in this article, we will refer to this directory as $ICU_SOURCE_DIR.
7. Extract the ICU compressed source code using the following command:
```
gzip –d icu4c-4_4_2-src.tgz
```
   You should get an archived file, icu4c-4_4_2-src.tar, after running the command.
8. Extract the archived file (icu4c-4_4_2-src.tar) using the following command:
```
unpax-icu.sh icu4c-4_4_2-src.tar
```
   It is very important to extract the archive using unpax-icu.sh provided with ICU, rather than the standard tar utilities. unpax-icu.sh can help to create files of the correct format that is specific to ICU. It will take about 40 minutes to extract the files depending on your environment. Here are the results:

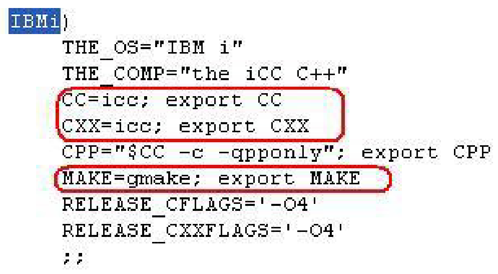
   Figure 6. Extracting the archived file using unpax-icu.sh



```
                              QSH Command Entry

 icu/source/data/curr/ha.txt
 icu/source/data/curr/sr.txt
 icu/source/extra/uconv/samples/utf8/utf-8-demo.txt
 icu/source/extra/uconv/resources/fr.txt
 icu/source/extra/uconv/resources/root.txt
 pax: 001-2298 For archive file icu4c-4_4_2-src.tar and volume 1, 3114 files w
 ere processed with 0 bytes read and 73049088 bytes written.

 Generating qsh compatible configure ...


 /tmp/zjj/icutemp/unpax-icu.sh has completed extracting ICU from icu4c-4_4_2-s
 rc.tar - 988 binary files extracted.
 $

===>
```

9. Modify runConfigureICU, located in $ICU_SOURCE_DIR/icu/source, as shown in Figure 7.

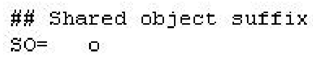
   Figure 7. Revise the configure file, runConfigureICU



```
IBMi)
    THE_OS="IBM i"
    THE_COMP="the iCC C++"
    CC=icc; export CC
    CXX=icc; export CXX
    CPP="$CC -c -qpponly"; export CPP
    MAKE=gmake; export MAKE
    RELEASE_CFLAGS='-O4'
    RELEASE_CXXFLAGS='-O4'
    ;;
```

10. Update mh-os400 in $ICU_SOURCE_DIR/icu/source/config/mh-os400, as shown in Figure 8.

Figure 8. Revise mh-os400

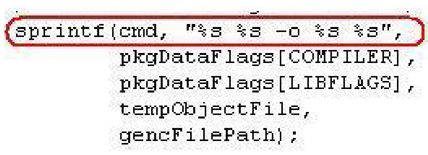


```
## Shared object suffix
SO=    o
```

This is because icc only accepts share libraries with the suffix "*.o".

11. Revise line number 1219 in the pkgdata.cpp file in $ICU_SOURCE_DIR/icu/source/tools/pkgdata by adding a comma, as shown in Figure 9. This is a syntax error in ICU4C4.4.2.

Figure 9. Revise pkgdata.cpp



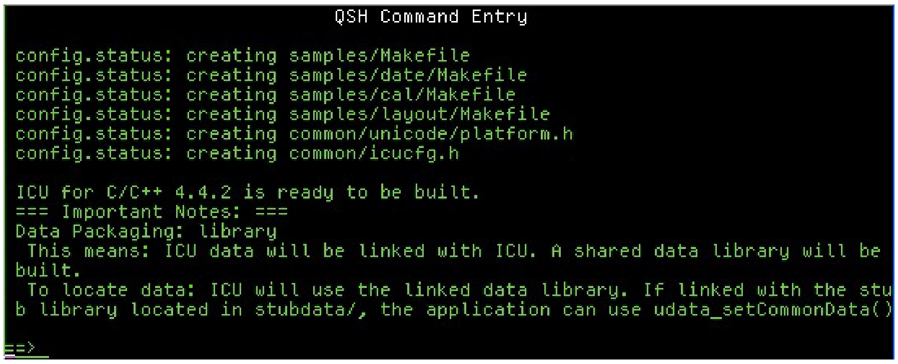
```
sprintf(cmd, "%s %s -o %s %s",
            pkgDataFlags[COMPILER],
            pkgDataFlags[LIBFLAGS],
            tempObjectFile,
            gencFilePath);
```

12. Run the following command to configure ICU and generate the necessary make files.

```
./runConfigureICU IBMi
```

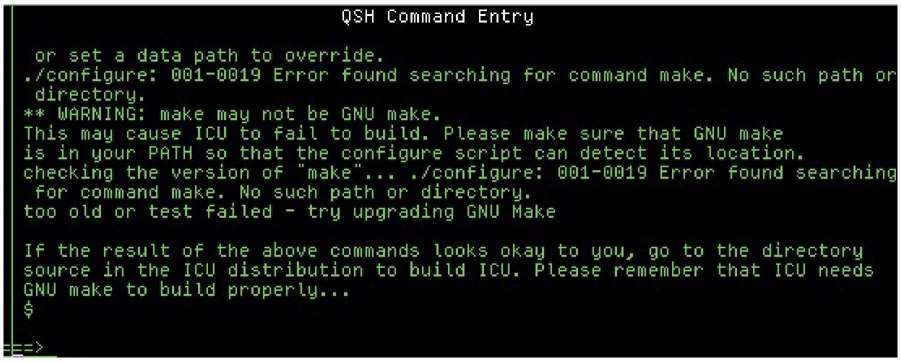
When finished, you will see the results as shown in Figure 10 and Figure 11. We can ignore the warning (as shown in Figure 11) and proceed further.

Figure 10. runConfigureICU result



```
                              QSH Command Entry

config.status: creating samples/Makefile
config.status: creating samples/date/Makefile
config.status: creating samples/cal/Makefile
config.status: creating samples/layout/Makefile
config.status: creating common/unicode/platform.h
config.status: creating common/icucfg.h

ICU for C/C++ 4.4.2 is ready to be built.
=== Important Notes: ===
Data Packaging: library
 This means: ICU data will be linked with ICU. A shared data library will be
built.
 To locate data: ICU will use the linked data library. If linked with the stu
b library located in stubdata/, the application can use udata_setCommonData()

==>
```

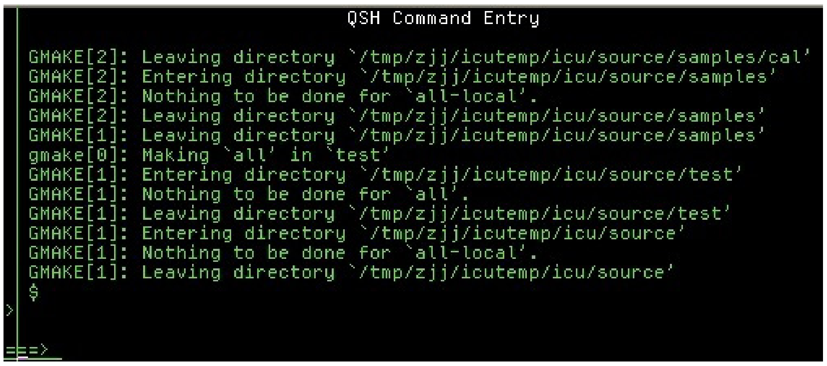
Figure 11. runConfigureICU warning



```
                              QSH Command Entry

 or set a data path to override.
./configure: 001-0019 Error found searching for command make. No such path or
 directory.
** WARNING: make may not be GNU make.
This may cause ICU to fail to build. Please make sure that GNU make
is in your PATH so that the configure script can detect its location.
checking the version of "make"... ./configure: 001-0019 Error found searching
 for command make. No such path or directory.
too old or test failed - try upgrading GNU Make

If the result of the above commands looks okay to you, go to the directory
source in the ICU distribution to build ICU. Please remember that ICU needs
GNU make to build properly...
$

==>
```

13. Type `gmake` to build ICU. After about 1 hour, gmake would complete and display a screen as shown Figure 12.

    Figure 12. Completion of building ICU

```
                             QSH Command Entry

 GMAKE[2]: Leaving directory `/tmp/zjj/icutemp/icu/source/samples/cal'
 GMAKE[2]: Entering directory `/tmp/zjj/icutemp/icu/source/samples'
 GMAKE[2]: Nothing to be done for `all-local'.
 GMAKE[2]: Leaving directory `/tmp/zjj/icutemp/icu/source/samples'
 GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source/samples'
 gmake[0]: Making `all' in `test'
 GMAKE[1]: Entering directory `/tmp/zjj/icutemp/icu/source/test'
 GMAKE[1]: Nothing to be done for `all'.
 GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source/test'
 GMAKE[1]: Entering directory `/tmp/zjj/icutemp/icu/source'
 GMAKE[1]: Nothing to be done for `all-local'.
 GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source'
 $
>

===>
```

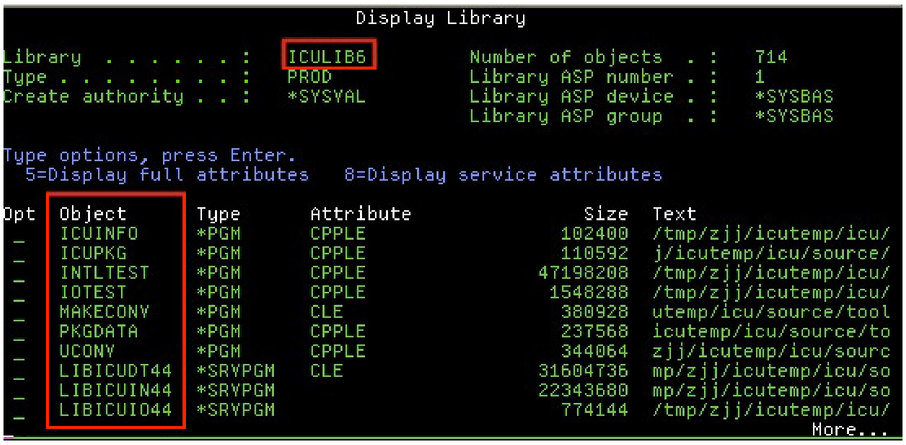
    Building ICU on IBM i is now complete.

Check the building result

In this section, we have listed three ways to check the building result.

1. The basic and simple way to check your results is by checking whether the files (service programs, programs, and modules) are created in the library you specified by adding the environment variable: OUTPUTDIR. In our case, it should be located in the ICULIB6 library, as shown in Figure 13.

    Figure 13. Check the build results in OUTPUTDIR

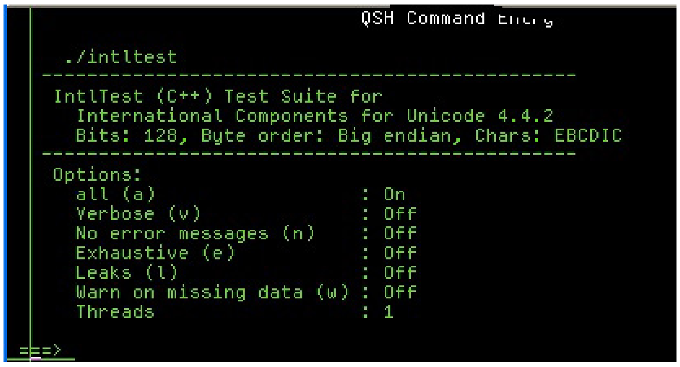
```
                              Display Library

Library . . . . . . :   ICULIB6          Number of objects  . :   714
Type . . . . . . . . :   PROD             Library ASP number . :   1
Create authority . . :   *SYSVAL          Library ASP device . :   *SYSBAS
                                          Library ASP group  . :   *SYSBAS

Type options, press Enter.
  5=Display full attributes   8=Display service attributes

Opt  Object      Type      Attribute              Size  Text
 _   ICUINFO     *PGM      CPPLE                102400  /tmp/zjj/icutemp/icu/
 _   ICUPKG      *PGM      CPPLE                110592  j/icutemp/icu/source/
 _   INTLTEST    *PGM      CPPLE              47198208  /tmp/zjj/icutemp/icu/
 _   IOTEST      *PGM      CPPLE               1548288  /tmp/zjj/icutemp/icu/
 _   MAKECONV    *PGM      CLE                  380928  utemp/icu/source/tool
 _   PKGDATA     *PGM      CPPLE                237568  icutemp/icu/source/to
 _   UCONV       *PGM      CPPLE                344064  zjj/icutemp/icu/sourc
 _   LIBICUDT44  *SRVPGM   CLE                31604736  mp/zjj/icutemp/icu/so
 _   LIBICUIN44  *SRVPGM                      22343680  mp/zjj/icutemp/icu/so
 _   LIBICUIO44  *SRVPGM                        774144  /tmp/zjj/icutemp/icu/
                                                                       More...
```

2. Check your results by running the following command as suggested by the ICU *readme* document.

```
gmake check QIBM_MULTI_THREADED=Y
```

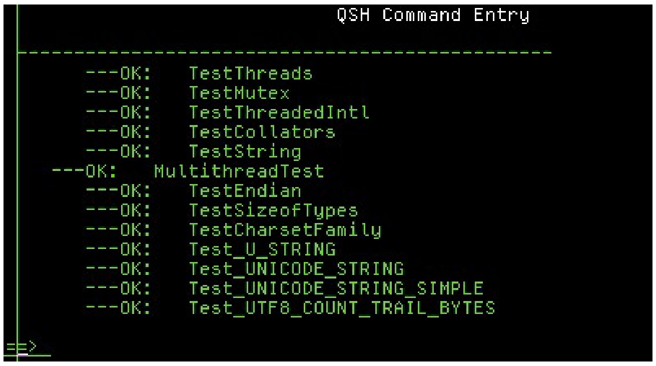
    This is not a mandatory step to build ICU. It is only used for testing your build results, and the results might depend on your build environment (compiler, OS version, and build options). Figure 14 and Figure 15 are our results for your reference.

Figure 14. gmake check results 1



```
                                QSH Command Entry

   ./intltest
 -----------------------------------------------
  IntlTest (C++) Test Suite for
    International Components for Unicode 4.4.2
    Bits: 128, Byte order: Big endian, Chars: EBCDIC
 -----------------------------------------------
  Options:
    all (a)                  : On
    Verbose (v)              : Off
    No error messages (n)    : Off
    Exhaustive (e)           : Off
    Leaks (l)                : Off
    Warn on missing data (w) : Off
    Threads                  : 1

 ===>
```

Figure 15. gmake check results 2



```
                                QSH Command Entry

 -----------------------------------------------
       ---OK:   TestThreads
       ---OK:   TestMutex
       ---OK:   TestThreadedIntl
       ---OK:   TestCollators
       ---OK:   TestString
    ---OK:   MultithreadTest
       ---OK:   TestEndian
       ---OK:   TestSizeofTypes
       ---OK:   TestCharsetFamily
       ---OK:   Test_U_STRING
       ---OK:   Test_UNICODE_STRING
       ---OK:   Test_UNICODE_STRING_SIMPLE
       ---OK:   Test_UTF8_COUNT_TRAIL_BYTES

 ===>
```

3. A sample program:
   Here is a sample program, icuDateTime.c that uses the ICU libraries we just built. The program will output the current date and time in the correct format for the locale and culture specified by the user.

Listing 1. icuDateTime.c source code

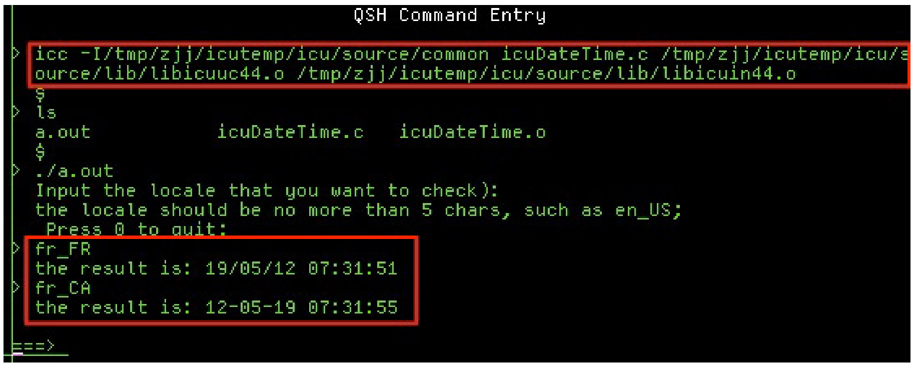
```
|--------10----20----30-----40----50----60----70---80-----|
/*
 * icuDateTime.c source code
 * This program is only for demonstrative purpose
 * to show writing globalized programs
 * using ICU APIs.
 * There are generally 2 steps for this program
 * Firstly, it gets the current system time.
 * Secondly, it displays the time in the format for the locale user specifies.
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "unicode/urename.h"
#include "unicode/utypes.h"
#include "unicode/udat.h"

/*
 * In this program, we use the lower-case two-letter
 * language codes defined in ISO-639
 * and uppper-case two-letter country codes
 * defined in ISO-3166 to represent a Locale.
 * For example, "en_US" represents a valid locale,
 * i.e. English language in United States.
 * So we define locale length to 5 letters.
 */
#define LOCALE_LEN 5

int main()
{
 UErrorCode status = U_ZERO_ERROR;
 UChar *u_strDateTime;
 char* strDateTime;
 int32_t nDateTimeLen = 0, i = 0, j = 0;
 UDate udDateTime = 0.0;
 char strLocale[LOCALE_LEN + 1] = {0};
 UDateFormat* dfmt;

 printf("Input the locale that you want to check):\n");
 printf("the locale should be no more than 5 chars,
 such as en_US;\n Press 0 to quit;\n");

 while ('0' != strLocale[0])
 {
 Memset#strLocale, 0, sizeof(strLocale)#;

 // Get user's input for locales
 gets(strLocale);

 // Get date and time format for the specified locale
 dfmt = udat_open(UDAT_MEDIUM, UDAT_SHORT, strLocale, NULL, -1, NULL, -1, &status);
 if(0 == dfmt)
 {
  printf("an error occurs. Error is %s\n", u_errorName(status));
 }

 // time(NULL) returns the current time by seconds,
        //which should be converted to milliseconds.
 udDateTime = 1000.0 * time(NULL);

 nDateTimeLen = 0;
 // The statement here is intended to get the output size needed.
 nDateTimeLen = udat_format(dfmt, udDateTime, NULL, nDateTimeLen, NULL, &status);
 if (status == U_BUFFER_OVERFLOW_ERROR)
 {
   status=U_ZERO_ERROR;
   u_strDateTime=(UChar*)malloc(sizeof(UChar) * (nDateTimeLen+1) );
   udat_format(dfmt, udDateTime, u_strDateTime, nDateTimeLen+1, NULL, &status);
   if(U_SUCCESS(status))
   {
  /*
   * Convert "UChar*" to "char*" to print it in console screen.
   * Generally speaking, 2 times buffer size of its wide-char counterpart
                * should be
   * enough for storing the resulting string.
   */
  strDateTime = (char*)malloc(2 * sizeof(char) * (nDateTimeLen + 1));
  u_austrcpy(strDateTime, u_strDateTime);
  printf("the result is: %s\n", strDateTime);

  free(u_strDateTime);
  free(strDateTime);

  continue;
    }

  free(u_strDateTime);
 }
 printf("an error occurs. Error is %s\n", u_errorName(status));
 }

 return 0;
}
```

Compile the above sample code using the following command:
`icc -I/tmp/zjj/icutemp/icu/source/common icuDateTime.c /tmp/zjj/icutemp/icu/source/lib/libicuuc44.o /tmp/zjj/icutemp/icu/source/lib/libicuin44.o`,
where **/tmp/zjj/icutemp/icu/source/common** specifies the location of the ICU header files that we used in this program, **/tmp/zjj/icutemp/icu/source/lib/libicuuc44.o** and **/tmp/zjj/icutemp/icu/source/lib/libicuin44.o** are the two ICU libraries we just built.
After compiling the program, try to run it using the following command:

```
./a.out
```

We can see that the date and time format are formatted according to the locales specified (refer to Figure 16).

Figure 16. Compile and run the ICU sample program



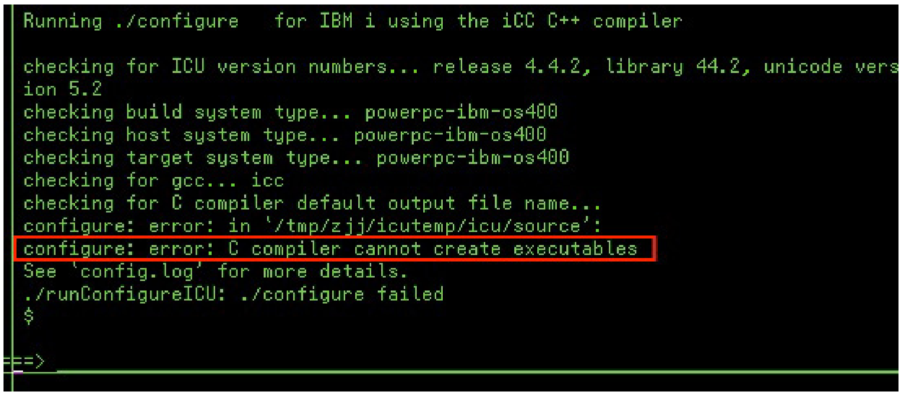
```
                              QSH Command Entry

> icc -I/tmp/zjj/icutemp/icu/source/common icuDateTime.c /tmp/zjj/icutemp/icu/s
  ource/lib/libicuuc44.o /tmp/zjj/icutemp/icu/source/lib/libicuin44.o
  $
> ls
  a.out           icuDateTime.c   icuDateTime.o
  $
> ./a.out
  Input the locale that you want to check):
  the locale should be no more than 5 chars, such as en_US;
   Press 0 to quit:
> fr_FR
  the result is: 19/05/12 07:31:51
> fr_CA
  the result is: 12-05-19 07:31:55

===>
```

Troubleshooting problems when building ICU on IBM i

Generally speaking, if you follow strictly the steps, you should be able to build ICU successfully. However, the contents in the following sections are helpful, if you encounter some problems.

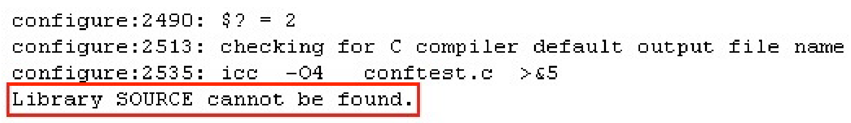
- ./runConfigureICU fails with error: The C compiler cannot create executables (as shown in Figure 17). When you encounter this error, you should check the config.log file in the $ICU_SOURCE_DIR/icu/source directory first. If you see errors such as `library XXXX cannot be found` in the log (as shown in Figure 18), it usually means that you cannot even compile any source code in this environment using icc. For example, when we compile a simple C source code, ctest.c, it can be compiled successfully on other platforms whereas it fails in this environment (as shown in Figure 19). We get similar errors as in the config.log file.

Figure 17. ICU configuration failure as icc cannot create executables



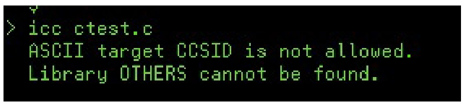
```
Running ./configure   for IBM i using the iCC C++ compiler

checking for ICU version numbers... release 4.4.2, library 44.2, unicode vers
ion 5.2
checking build system type... powerpc-ibm-os400
checking host system type... powerpc-ibm-os400
checking target system type... powerpc-ibm-os400
checking for gcc... icc
checking for C compiler default output file name...
configure: error: in `/tmp/zjj/icutemp/icu/source':
configure: error: C compiler cannot create executables
See `config.log' for more details.
./runConfigureICU: ./configure failed
$

==>
```

Figure 18. library XXXX cannot be found error



```
configure:2490: $? = 2
configure:2513: checking for C compiler default output file name
configure:2535: icc  -O4   conftest.c  >&5
Library SOURCE cannot be found.
```

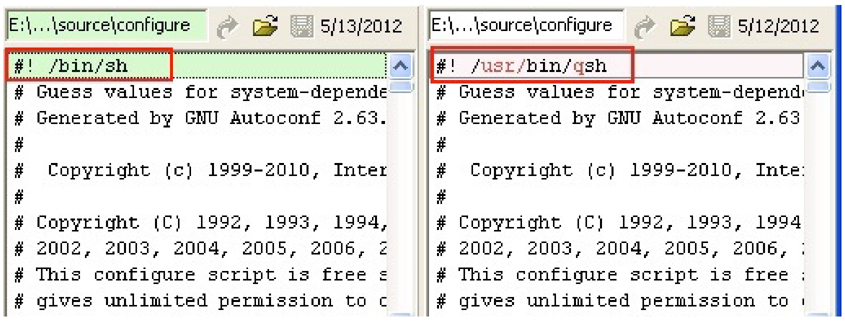
Figure 19. icc not working as expected



```
> icc ctest.c
  ASCII target CCSID is not allowed.
  Library OTHERS cannot be found.
```

In another case, you might not see the error as shown in Figure 18, and see only the errors as shown in Figure 17 in the log file. In such cases, you can compile the C source code successfully using icc. It is usually caused by an uncustomized configuration file for IBM i. There is a simple way to check if your configuration file is customized for IBM i. Check the header of the file to see whether the configuration will use qsh or sh to run. If the header shows qsh (as shown in the right pane of Figure 20), then this file should already be customized for IBM i.

Figure 20. Header of the two different configuration files



```
E:\...\source\configure   5/13/2012
#! /bin/sh
# Guess values for system-depende
# Generated by GNU Autoconf 2.63.
#
#   Copyright (c) 1999-2010, Inter
#
# Copyright (C) 1992, 1993, 1994,
# 2002, 2003, 2004, 2005, 2006, 2
# This configure script is free s
# gives unlimited permission to c
```

```
E:\...\source\configure   5/12/2012
#! /usr/bin/qsh
# Guess values for system-depend
# Generated by GNU Autoconf 2.63
#
#   Copyright (c) 1999-2010, Inte
#
# Copyright (C) 1992, 1993, 1994
# 2002, 2003, 2004, 2005, 2006,
# This configure script is free
# gives unlimited permission to
```

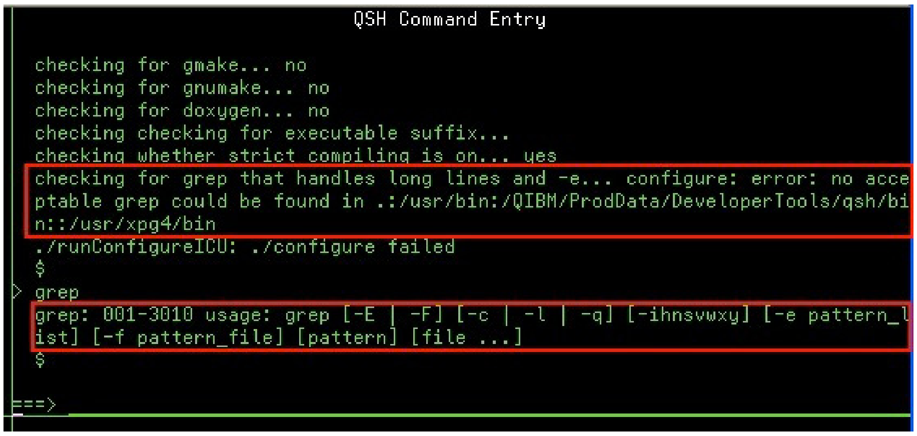
Resolution: For the first case, such an error occurs usually because you do not set the environment variable, OUTPUTDIR, when using icc. Export the variable in either IBM OS/400® Command Entry (recommended) or QSH Command Entry to resolve the problem. Here is an example:

`addenvvar envvar(OUTPUTDIR) value('iculib6') replace(*YES)` in OS's Command Entry, or `export OUTPUTDIR=iculib6` in QSH Command Entry.

For the second case, using the `unpax-icu.sh icu4c-4_4_2-src.tar` command to extract the source package will help to generate a customized configuration file for IBM i.

- ./runConfigureICU fails with error: No acceptable XXXX (stands for some common command, such as grep, ar, awk and so on) can be found in *SOMEWHERE* (location). Figure 21 indicates an instance where grep cannot be found in the environment. When you type **grep** in the command line, the results show that the command can be located in the environment. Actually, grep is located in /QOpenSys/usr/bin.

Figure 21. grep command cannot be found



```
                         QSH Command Entry

 checking for gmake... no
 checking for gnumake... no
 checking for doxygen... no
 checking checking for executable suffix...
 checking whether strict compiling is on... yes
 checking for grep that handles long lines and -e... configure: error: no acce
 ptable grep could be found in .:/usr/bin:/QIBM/ProdData/DeveloperTools/qsh/bi
 n::/usr/xpg4/bin
 ./runConfigureICU: ./configure failed
 $
> grep
 grep: 001-3010 usage: grep [-E | -F] [-c | -l | -q] [-ihnsvwxy] [-e pattern_l
 ist] [-f pattern_file] [pattern] [file ...]
 $

===>
```

Resolution: Add the environment variable of the command or revise the PATH variable in either Command Entry or QSH Command Entry. Here are some examples:

`adenvvar envvar(GREP) value('grep') replace(*YES)`, or

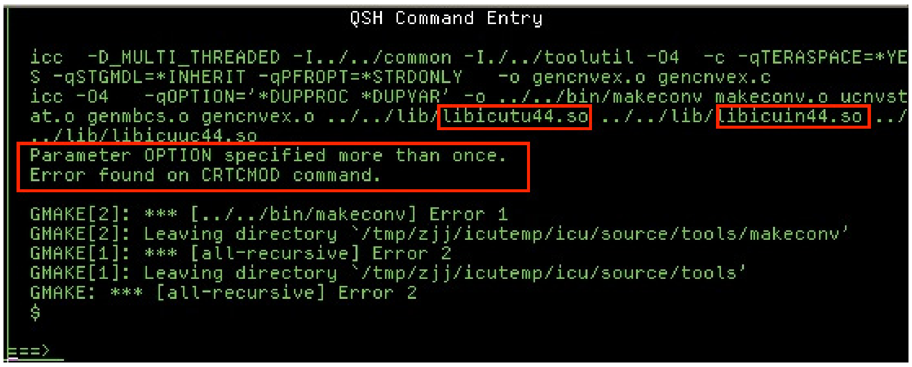
`chgenvvar envvar(PATH) value('/QIBM/ProdData/DeveloperTools/qsh/bin:/usr/bin:.:/QOpenSys/usr/bin')`, or

`export PATH=/QIBM/ProdData/DeveloperTools/qsh/bin:$PATH`, or

`export GREP=grep`.

- gmake fails with error: This is an error found on the CRTMOD command, as shown in Figure 22. When examining the compilation command, we find that all the shared objects have a suffix **.so**, as shown in Figure 22. This is true on Linux while not on IBM i. Specifically, in the environment of Tools for Developer on IBM i, shared objects, otherwise called **service program**, must have a suffix ***.o** to compile the code successfully using icc.

Figure 22. Error with the CRTCMOD command

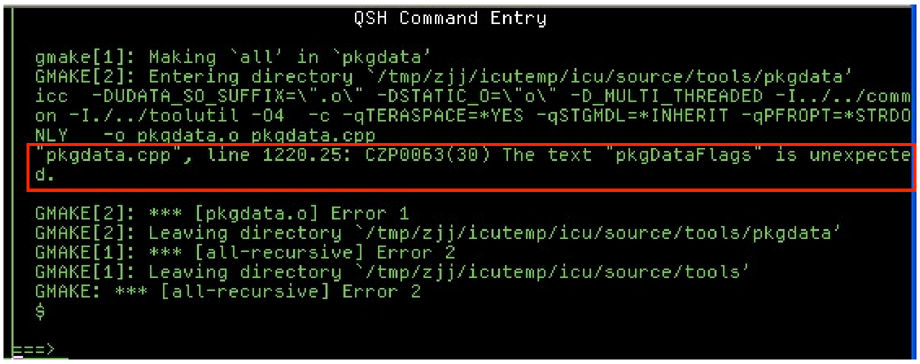


```
                              QSH Command Entry

  icc  -D_MULTI_THREADED -I../../common -I./../toolutil -O4  -c -qTERASPACE=*YE
  S -qSTGMDL=*INHERIT -qPFROPT=*STRDONLY   -o gencnvex.o gencnvex.c
  icc -O4   -qOPTION='*DUPPROC *DUPVAR' -o ../../bin/makeconv makeconv.o ucnvst
  at.o genmbcs.o gencnvex.o ../../lib/libicutu44.so ../../lib/libicuin44.so ../
  ../lib/libicuuc44.so
  Parameter OPTION specified more than once.
  Error found on CRTCMOD command.

  GMAKE[2]: *** [../../bin/makeconv] Error 1
  GMAKE[2]: Leaving directory `/tmp/zjj/icutemp/icu/source/tools/makeconv'
  GMAKE[1]: *** [all-recursive] Error 2
  GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source/tools'
  GMAKE: *** [all-recursive] Error 2
  $

===>
```

Resolution: Revise the mh-os400 file located in the $ICU_SOURCE_DIR/icu/source/config directory as instructed in the previous section and rebuild the ICU source code.

- The source file, pkgdata.cpp, located in the $ICU_SOURCE_DIR/icu/source/tools/pkgdata directory cannot be compiled successfully. The error, **The text "pkgDataFlags" is unexpected** is displayed.

Figure 23. Error when compiling pkgdata.cpp

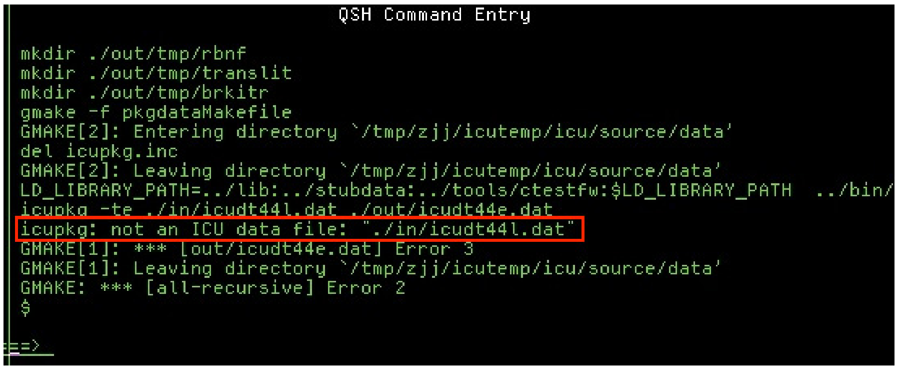


```
                              QSH Command Entry

  gmake[1]: Making `all' in `pkgdata'
  GMAKE[2]: Entering directory `/tmp/zjj/icutemp/icu/source/tools/pkgdata'
  icc  -DUDATA_SO_SUFFIX=\".o\" -DSTATIC_O=\"o\" -D_MULTI_THREADED -I../../comm
  on -I./../toolutil -O4  -c -qTERASPACE=*YES -qSTGMDL=*INHERIT -qPFROPT=*STRDO
  NLY   -o pkgdata.o pkgdata.cpp
  "pkgdata.cpp", line 1220.25: CZP0063(30) The text "pkgDataFlags" is unexpecte
  d.

  GMAKE[2]: *** [pkgdata.o] Error 1
  GMAKE[2]: Leaving directory `/tmp/zjj/icutemp/icu/source/tools/pkgdata'
  GMAKE[1]: *** [all-recursive] Error 2
  GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source/tools'
  GMAKE: *** [all-recursive] Error 2
  $

===>
```

Resolution: This is due to a syntax error in the pkgdata.cpp file. Revise the file as shown in Figure 9 and recompile the ICU code.

- Make ICU source code fails with an error, **not an ICU data file: './in/icudt44l.dat'**.Tue icudt44.dat file is provided with the ICU source code, which is located in the $ICU_SOURCE_DIR/icu/source/data/in directory. It is used to build the ICU data files, such as icudt44e.dat (in this case), as shown in Figure 24. Theoretically, the file should be accepted by the utility icupkg, because it is provided together with the ICU source code. The error is usually caused by incorrect ways of unpacking the ICU source code in tar format. For example, you might unpack the tar file directly using the tar command on the qsh environment on IBM i, or upload the source code files after unpacking the tar file on other platforms.

Figure 24. Not an ICU data file error

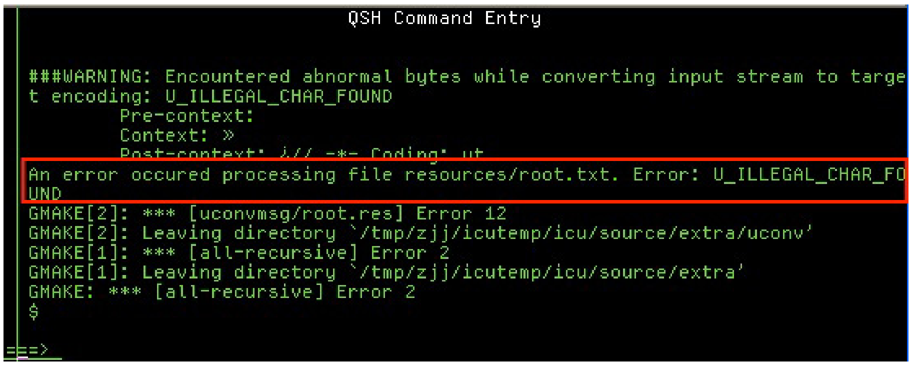


```
                              QSH Command Entry

 mkdir ./out/tmp/rbnf
 mkdir ./out/tmp/translit
 mkdir ./out/tmp/brkitr
 gmake -f pkgdataMakefile
 GMAKE[2]: Entering directory `/tmp/zjj/icutemp/icu/source/data'
 del icupkg.inc
 GMAKE[2]: Leaving directory `/tmp/zjj/icutemp/icu/source/data'
 LD_LIBRARY_PATH=../lib:../stubdata:../tools/ctestfw:$LD_LIBRARY_PATH  ../bin/
 icupkg -te ./in/icudt44l.dat ./out/icudt44e.dat
 icupkg: not an ICU data file: "./in/icudt44l.dat"
 GMAKE[1]: *** [out/icudt44e.dat] Error 3
 GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source/data'
 GMAKE: *** [all-recursive] Error 2
 $

===>
```

Resolution: Use the `unpax-icu.sh icu4c-4_4_2-src.tar` command to extract the source package rather than any other methods.

- Make ICU code fails with the **U_ILLEGAL_CHAR_FOUND** error, as shown in Figure 25. This might again be due to incorrectly unpacking the TAR file, which is the same problem as shown in Figure 24.

Figure 25. U_ILLEGAL_CHAR_FOUND error



```
                              QSH Command Entry

 ###WARNING: Encountered abnormal bytes while converting input stream to targe
 t encoding: U_ILLEGAL_CHAR_FOUND
         Pre-context:
         Context: »
         Post-context: ¿// -*- Coding: ut
 An error occured processing file resources/root.txt. Error: U_ILLEGAL_CHAR_FO
 UND
 GMAKE[2]: *** [uconvmsg/root.res] Error 12
 GMAKE[2]: Leaving directory `/tmp/zjj/icutemp/icu/source/extra/uconv'
 GMAKE[1]: *** [all-recursive] Error 2
 GMAKE[1]: Leaving directory `/tmp/zjj/icutemp/icu/source/extra'
 GMAKE: *** [all-recursive] Error 2
 $

===>
```

Resolution: Use the `unpax-icu.sh icu4c-4_4_2-src.tar` command to extract the source package rather than any other methods.

Summary

In this article, we provided step-by-step instructions to build ICU4.4.2 on IBM i, including preparing, building the environment, and verifying the build results. We summarized the common issues that might be encountered during the process and suggested the possible resolutions. ICU libraries can definitely help to develop globalized applications.

Resources

Here are a few materials that would help you get a better understanding of this article:

- ICU 4.4.2 can be download from the ICU download page
- For more information about IBM Tools for Developers for IBM i, refer to IBM PartnerWorld

- If you encounter other common ICU problems that are not listed in this article, then ICU FAQs is a good place to start with.
- You can check the sample program's result with ICU4C demonstrations: Locale Explorer.
- For CCSID information on the IBM i system, refer to the IBM i information center.