Internationalization with gettext

Supporting multiple natural languages in a user interface

Bastiaan Veelo, PhD.   DConf 2023

Previously…

Previously…

Coming from Pascal: home grown system

- Translate the source code

Previously…

Coming from Pascal: home grown system

- Translate the source code

Disadvantages:

- Badly scalable. Need to compile for every language.
- Not user friendly. Users only order one or several specific languages, need to change executables to change language.
- Hard to maintain — translation rot / no fallback / no “needs work” markers
- No run-time variations (plural variants)
- No equivalent for “format”:
  - Translators work with fragments, lack context
  - Translators cannot swap arguments

Previously…

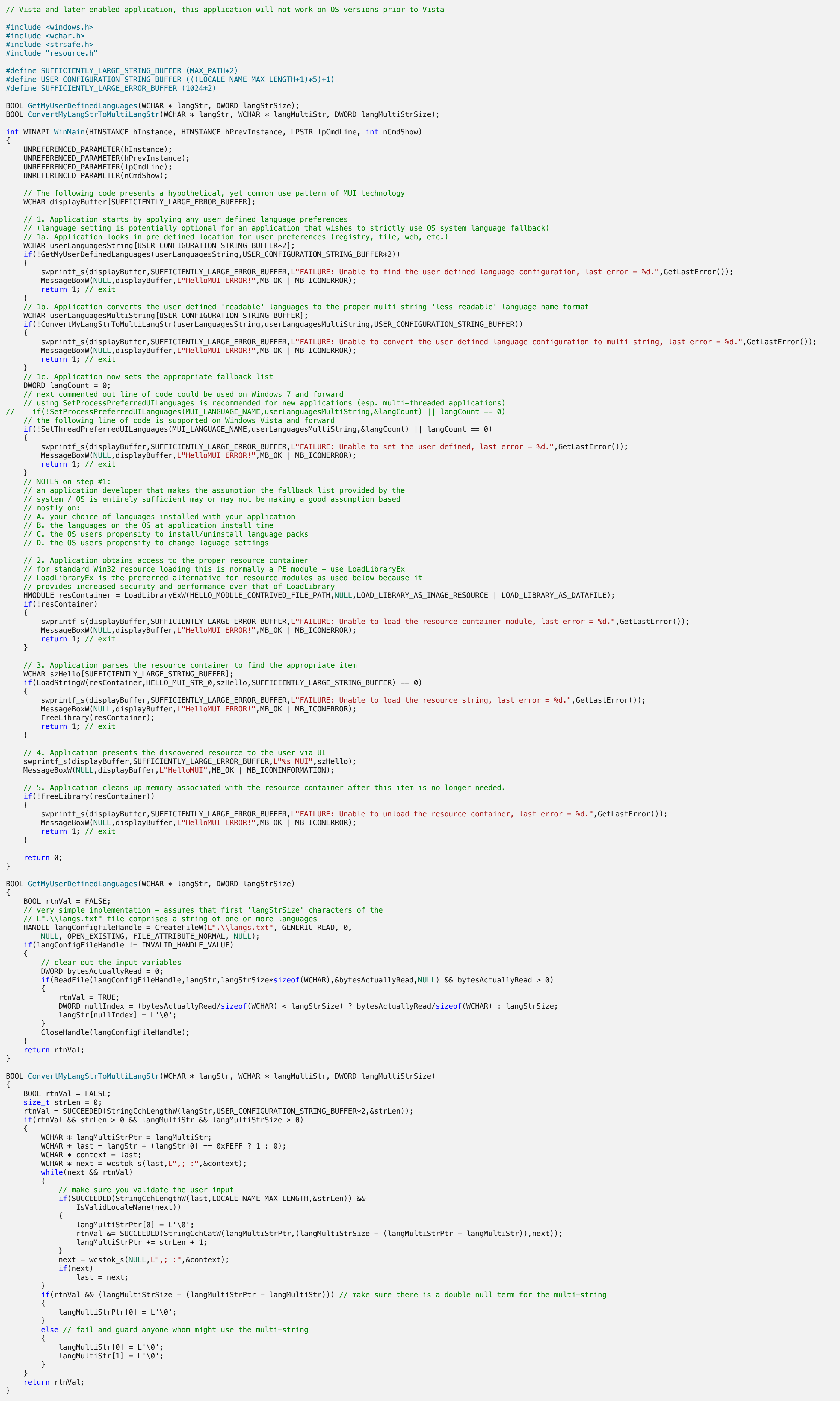
Coming from Pascal: home grown system

- Translate the source code

Disadvantages:

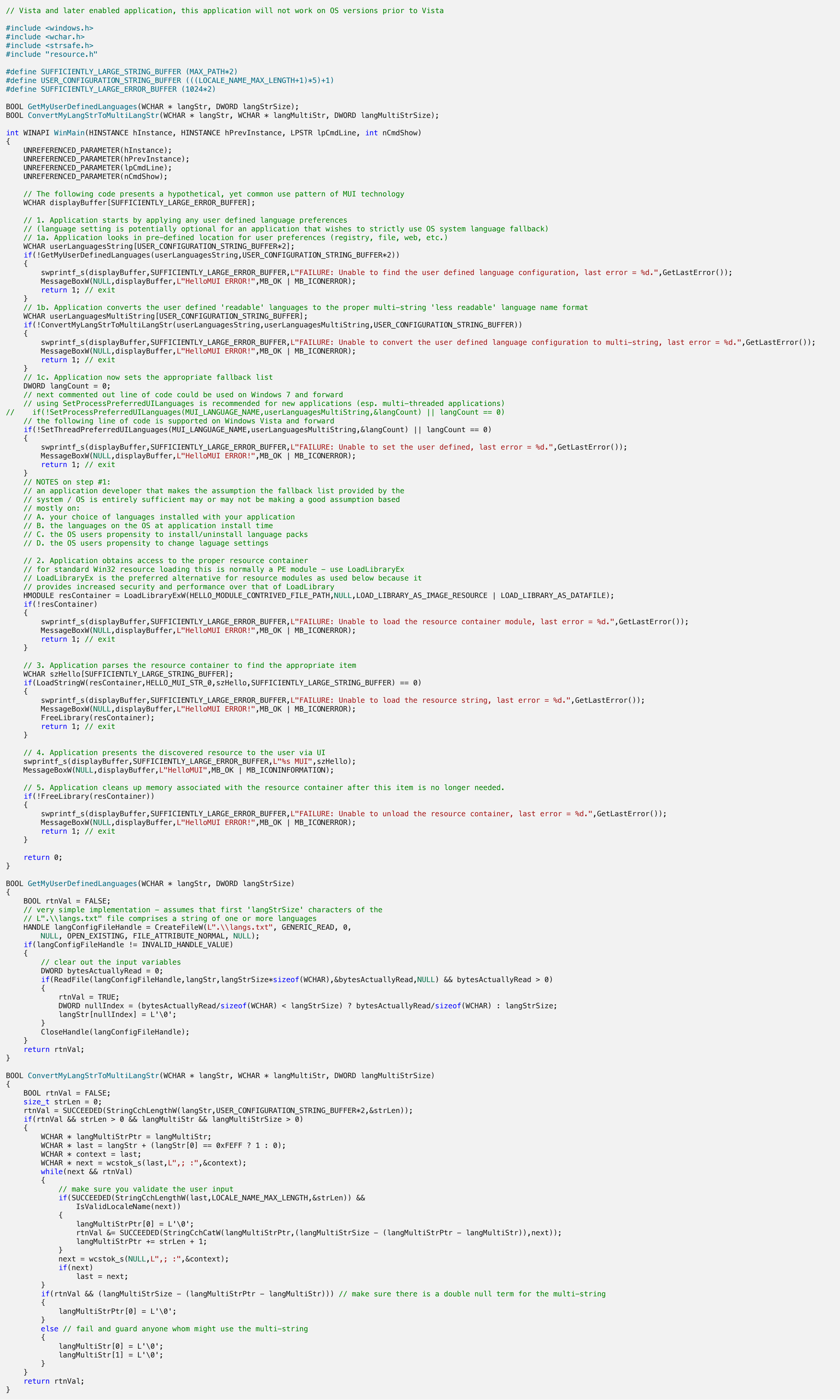
- Badly scalable. Need to compile for every language.
- Not user friendly. Users only order one or several specific languages, need to change executables to change language.
- Hard to maintain — translation rot / no fallback / no “needs work” markers
- No run-time variations (plural variants)
- No equivalent for “format”:
    - Translators work with fragments, lack context
    - Translators cannot swap arguments

Transcompilation = opportunity to switch. What is best?



```c
// Vista and later enabled application, this application will not work on OS versions prior to Vista

#include <windows.h>
#include <wchar.h>
#include <strsafe.h>
#include "resource.h"

#define SUFFICIENTLY_LARGE_STRING_BUFFER (MAX_PATH*2)
#define USER_CONFIGURATION_STRING_BUFFER (((LOCALE_NAME_MAX_LENGTH+1)*5)+1)
#define SUFFICIENTLY_LARGE_ERROR_BUFFER (1024*2)

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize);
BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    UNREFERENCED_PARAMETER(hInstance);
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);
    UNREFERENCED_PARAMETER(nCmdShow);

    // The following code presents a hypothetical, yet common use pattern of MUI technology
    WCHAR displayBuffer[SUFFICIENTLY_LARGE_ERROR_BUFFER];

    // 1. Application starts by applying any user defined language preferences
    // (language setting is potentially optional for an application that wishes to strictly use OS system language fallback)
    // 1a. Application looks in pre-defined location for user preferences (registry, file, web, etc.)
    WCHAR userLanguagesString[USER_CONFIGURATION_STRING_BUFFER*2];
    if(!GetMyUserDefinedLanguages(userLanguagesString,USER_CONFIGURATION_STRING_BUFFER*2))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to find the user defined language configuration, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // 1b. Application converts the user defined 'readable' languages to the proper multi-string 'less readable' language name format
    WCHAR userLanguagesMultiString[USER_CONFIGURATION_STRING_BUFFER];
    if(!ConvertMyLangStrToMultiLangStr(userLanguagesString,userLanguagesMultiString,USER_CONFIGURATION_STRING_BUFFER))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to convert the user defined language configuration to multi-string, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // 1c. Application now sets the appropriate fallback list
    DWORD langCount = 0;
    // next commented out line of code could be used on Windows 7 and forward
    // using SetProcessPreferredUILanguages is recommended for new applications (esp. multi-threaded applications)
//    if(!SetProcessPreferredUILanguages(MUI_LANGUAGE_NAME,userLanguagesMultiString,&langCount) || langCount == 0)
    // the following line of code is supported on Windows Vista and forward
    if(!SetThreadPreferredUILanguages(MUI_LANGUAGE_NAME,userLanguagesMultiString,&langCount) || langCount == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to set the user defined, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // NOTES on step #1:
    // an application developer that makes the assumption the fallback list provided by the
    // system / OS is entirely sufficient may or may not be making a good assumption based
    // mostly on:
    // A. your choice of languages installed with your application
    // B. the languages on the OS at application install time
    // C. the OS users propensity to install/uninstall language packs
    // D. the OS users propensity to change laguage settings

    // 2. Application obtains access to the proper resource container
    // for standard Win32 resource loading this is normally a PE module - use LoadLibraryEx
    // LoadLibraryEx is the preferred alternative for resource modules as used below because it
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESOURCE | LOAD_LIBRARY_AS_DATAFILE);
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource container module, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource string, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer needed.
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource container, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    return 0;
}

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize)
{
    BOOL rtnVal = FALSE;
    // very simple implementation - assumes that first 'langStrSize' characters of the
    // L".\\langs.txt" file comprises a string of one or more languages
    HANDLE langConfigFileHandle = CreateFileW(L".\\langs.txt", GENERIC_READ, 0,
        NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if(langConfigFileHandle != INVALID_HANDLE_VALUE)
    {
        // clear out the input variables
        DWORD bytesActuallyRead = 0;
        if(ReadFile(langConfigFileHandle,langStr,langStrSize*sizeof(WCHAR),&bytesActuallyRead,NULL) && bytesActuallyRead > 0)
        {
            rtnVal = TRUE;
            DWORD nullIndex = (bytesActuallyRead/sizeof(WCHAR) < langStrSize) ? bytesActuallyRead/sizeof(WCHAR) : langStrSize;
            langStr[nullIndex] = L'\0';
        }
        CloseHandle(langConfigFileHandle);
    }
    return rtnVal;
}

BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize)
{
    BOOL rtnVal = FALSE;
    size_t strLen = 0;
    rtnVal = SUCCEEDED(StringCchLengthW(langStr,USER_CONFIGURATION_STRING_BUFFER*2,&strLen));
    if(rtnVal && strLen > 0 && langMultiStr && langMultiStrSize > 0)
    {
        WCHAR * langMultiStrPtr = langMultiStr;
        WCHAR * last = langStr + (langStr[0] == 0xFEFF ? 1 : 0);
        WCHAR * context = last;
        WCHAR * next = wcstok_s(last,L",; :",&context);
        while(next && rtnVal)
        {
            // make sure you validate the user input
            if(SUCCEEDED(StringCchLengthW(last,LOCALE_NAME_MAX_LENGTH,&strLen)) &&
                IsValidLocaleName(next))
            {
                langMultiStrPtr[0] = L'\0';
                rtnVal &= SUCCEEDED(StringCchCatW(langMultiStrPtr,(langMultiStrSize - (langMultiStrPtr - langMultiStr)),next));
                langMultiStrPtr += strLen + 1;
            }
            next = wcstok_s(NULL,L",; :",&context);
            if(next)
                last = next;
        }
        if(rtnVal && (langMultiStrSize - (langMultiStrPtr - langMultiStr))) // make sure there is a double null term for the multi-string
        {
            langMultiStrPtr[0] = L'\0';
        }
        else // fail and guard anyone whom might use the multi-string
        {
            langMultiStr[0] = L'\0';
            langMultiStr[1] = L'\0';
        }
    }
    return rtnVal;
}
```

https://learn.microsoft.com/en-us/windows/win32/intl/mui-application-specific-settings-sample-vista



```c
// Vista and later enabled application, this application will not work on OS versions prior to Vista

#include <windows.h>
#include <wchar.h>
#include <strsafe.h>
#include "resource.h"

#define SUFFICIENTLY_LARGE_STRING_BUFFER (MAX_PATH*2)
#define USER_CONFIGURATION_STRING_BUFFER (((LOCALE_NAME_MAX_LENGTH+1)*5)+1)
#define SUFFICIENTLY_LARGE_ERROR_BUFFER (1024*2)

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize);
BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    UNREFERENCED_PARAMETER(hInstance);
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);
    UNREFERENCED_PARAMETER(nCmdShow);

    // The following code presents a hypothetical, yet common use pattern of MUI technology
    WCHAR displayBuffer[SUFFICIENTLY_LARGE_ERROR_BUFFER];

    // 1. Application starts by applying any user defined language preferences
    // (language setting is potentially optional for an application that wishes to strictly use OS system language fallback)
    // 1a. Application looks in pre-defined location for user preferences (registry, file, web, etc.)
    WCHAR userLanguagesString[USER_CONFIGURATION_STRING_BUFFER*2];
    if(!GetMyUserDefinedLanguages(userLanguagesString,USER_CONFIGURATION_STRING_BUFFER*2))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to find the user defined language configuration, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // 1b. Application converts the user defined 'readable' languages to the proper multi-string 'less readable' language name format
    WCHAR userLanguagesMultiString[USER_CONFIGURATION_STRING_BUFFER];
    if(!ConvertMyLangStrToMultiLangStr(userLanguagesString,userLanguagesMultiString,USER_CONFIGURATION_STRING_BUFFER))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to convert the user defined language configuration to multi-string, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // 1c. Application now sets the appropriate fallback list
    DWORD langCount = 0;
    // next commented out line of code could be used on Windows 7 and forward
    // using SetProcessPreferredUILanguages is recommended for new applications (esp. multi-threaded applications)
//    if(!SetProcessPreferredUILanguages(MUI_LANGUAGE_NAME,userLanguagesMultiString,&langCount) || langCount == 0)
    // the following line of code is supported on Windows Vista and forward
    if(!SetThreadPreferredUILanguages(MUI_LANGUAGE_NAME,userLanguagesMultiString,&langCount) || langCount == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to set the user defined, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // NOTES on step #1:
    // an application developer that makes the assumption the fallback list provided by the
    // system / OS is entirely sufficient may or may not be making a good assumption based
    // mostly on:
    // A. your choice of languages installed with your application
    // B. the languages on the OS at application install time
    // C. the OS users propensity to install/uninstall language packs
    // D. the OS users propensity to change laguage settings

    // 2. Application obtains access to the proper resource container
    // for standard Win32 resource loading this is normally a PE module - use LoadLibraryEx
    // LoadLibraryEx is the preferred alternative for resource modules as used below because it
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESOURCE | LOAD_LIBRARY_AS_DATAFILE);
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource container module, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource string, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer needed.
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource container, last error = %d.",GetLastError());
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    return 0;
}

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize)
{
    BOOL rtnVal = FALSE;
    // very simple implementation - assumes that first 'langStrSize' characters of the
    // L".\\langs.txt" file comprises a string of one or more languages
    HANDLE langConfigFileHandle = CreateFileW(L".\\langs.txt", GENERIC_READ, 0,
        NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if(langConfigFileHandle != INVALID_HANDLE_VALUE)
    {
        // clear out the input variables
        DWORD bytesActuallyRead = 0;
        if(ReadFile(langConfigFileHandle,langStr,langStrSize*sizeof(WCHAR),&bytesActuallyRead,NULL) && bytesActuallyRead > 0)
        {
            rtnVal = TRUE;
            DWORD nullIndex = (bytesActuallyRead/sizeof(WCHAR) < langStrSize) ? bytesActuallyRead/sizeof(WCHAR) : langStrSize;
            langStr[nullIndex] = L'\0';
        }
        CloseHandle(langConfigFileHandle);
    }
    return rtnVal;
}

BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize)
{
    BOOL rtnVal = FALSE;
    size_t strLen = 0;
    rtnVal = SUCCEEDED(StringCchLengthW(langStr,USER_CONFIGURATION_STRING_BUFFER*2,&strLen));
    if(rtnVal && strLen > 0 && langMultiStr && langMultiStrSize > 0)
    {
        WCHAR * langMultiStrPtr = langMultiStr;
        WCHAR * last = langStr + (langStr[0] == 0xFEFF ? 1 : 0);
        WCHAR * context = last;
        WCHAR * next = wcstok_s(last,L",; :",&context);
        while(next && rtnVal)
        {
            // make sure you validate the user input
            if(SUCCEEDED(StringCchLengthW(last,LOCALE_NAME_MAX_LENGTH,&strLen)) &&
                IsValidLocaleName(next))
            {
                langMultiStrPtr[0] = L'\0';
                rtnVal &= SUCCEEDED(StringCchCatW(langMultiStrPtr,(langMultiStrSize - (langMultiStrPtr - langMultiStr)),next));
                langMultiStrPtr += strLen + 1;
            }
            next = wcstok_s(NULL,L",; :",&context);
            if(next)
                last = next;
        }
        if(rtnVal && (langMultiStrSize - (langMultiStrPtr - langMultiStr))) // make sure there is a double null term for the multi-string
        {
            langMultiStrPtr[0] = L'\0';
        }
        else // fail and guard anyone whom might use the multi-string
        {
            langMultiStr[0] = L'\0';
            langMultiStr[1] = L'\0';
        }
    }
    return rtnVal;
}
```

```
// Vista and later enabled application, this application will not work on OS versions prior to Vista

#include <windows.h>
#include <wchar.h>
#include <strsafe.h>
#include "resource.h"

#define SUFFICIENTLY_LARGE_STRING_BUFFER (MAX_PATH*2)
#define USER_CONFIGURATION_STRING_BUFFER (((LOCALE_NAME_MAX_LENGTH+1)*5)+1)
#define SUFFICIENTLY_LARGE_ERROR_BUFFER (1024*2)

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize);
BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    UNREFERENCED_PARAMETER(hInstance);
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);
    UNREFERENCED_PARAMETER(nCmdShow);

    // The following code presents a hypothetical, yet common use pattern of MUI technology
    WCHAR displayBuffer[SUFFICIENTLY_LARGE_ERROR_BUFFER];

    // 1. Application starts by applying any user defined language preferences
    // (language setting is potentially optional for an application that wishes to strictly use OS system l
    // 1a. Application looks in pre-defined location for user preferences (registry, file, web, etc.)
    WCHAR userLanguagesString[USER_CONFIGURATION_STRING_BUFFER*2];
    if(!GetMyUserDefinedLanguages(userLanguagesString,USER_CONFIGURATION_STRING_BUFFER*2))
    {
```

```
// Vista and later enabled application, this application will not work on OS versions prior to Vista

#include <windows.h>
#include <wchar.h>
#include <strsafe.h>
#include "resource.h"

#define SUFFICIENTLY_LARGE_STRING_BUFFER (MAX_PATH*2)
#define USER_CONFIGURATION_STRING_BUFFER (((LOCALE_NAME_MAX_LENGTH+1)*5)+1)
#define SUFFICIENTLY_LARGE_ERROR_BUFFER (1024*2)

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize);
BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    UNREFERENCED_PARAMETER(hInstance);
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);
    UNREFERENCED_PARAMETER(nCmdShow);

    // The following code presents a hypothetical, yet common use pattern of MUI technology
    WCHAR displayBuffer[SUFFICIENTLY_LARGE_ERROR_BUFFER];

    // 1. Application starts by applying any user defined language preferences
    // (language setting is potentially optional for an application that wishes to strictly use OS system
    // 1a. Application looks in pre-defined location for user preferences (registry, file, web, etc.)
    WCHAR userLanguagesString[USER_CONFIGURATION_STRING_BUFFER*2];
    if(!GetMyUserDefinedLanguages(userLanguagesString,USER_CONFIGURATION_STRING_BUFFER*2))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to find the user define
// Vista and later enabled application, this application will not work on OS versions prior to Vista

#include <windows.h>
#include <wchar.h>
#include <strsafe.h>
#include "resource.h"

#define SUFFICIENTLY_LARGE_STRING_BUFFER (MAX_PATH*2)
#define USER_CONFIGURATION_STRING_BUFFER (((LOCALE_NAME_MAX_LENGTH+1)*5)+1)
#define SUFFICIENTLY_LARGE_ERROR_BUFFER (1024*2)

BOOL GetMyUserDefinedLanguages(WCHAR * langStr, DWORD langStrSize);
BOOL ConvertMyLangStrToMultiLangStr(WCHAR * langStr, WCHAR * langMultiStr, DWORD langMultiStrSize);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    UNREFERENCED_PARAMETER(hInstance);
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);
    UNREFERENCED_PARAMETER(nCmdShow);

    // The following code presents a hypothetical, yet common use pattern of MUI technology
    WCHAR displayBuffer[SUFFICIENTLY_LARGE_ERROR_BUFFER];

    // 1. Application starts by applying any user defined language preferences
    // (language setting is potentially optional for an application that wishes to strictly use OS system
    // 1a. Application looks in pre-defined location for user preferences (registry, file, web, etc.)
    WCHAR userLanguagesString[USER_CONFIGURATION_STRING_BUFFER*2];
    if(!GetMyUserDefinedLanguages(userLanguagesString,USER_CONFIGURATION_STRING_BUFFER*2))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to find the user define
    {
        WCHAR * langMultiStrPtr = langMultiStr;
        WCHAR * last = langStr + (langStr[0] == 0xFEFF ? 1 : 0);
        WCHAR * context = last;
        WCHAR * next = wcstok_s(last,L",; :",&context);
        while(next && rtnVal)
        {
            // make sure you validate the user input
            if(SUCCEEDED(StringCchLengthW(last,LOCALE_NAME_MAX_LENGTH,&strLen)) &&
                IsValidLocaleName(next))
            {
                langMultiStrPtr[0] = L'\0';
                rtnVal &= SUCCEEDED(StringCchCatW(langMultiStrPtr,(langMultiStrSize - (langMultiStrPtr - l
                langMultiStrPtr += strLen + 1;
            }
            next = wcstok_s(NULL,L",; :",&context);
            if(next)
                last = next;
        }
        if(rtnVal && (langMultiStrSize - (langMultiStrPtr - langMultiStr))) // make sure there is a double
        {
            langMultiStrPtr[0] = L'\0';
        }
        else // fail and guard anyone whom might use the multi-string
        {
            langMultiStr[0] = L'\0';
            langMultiStr[1] = L'\0';
        }
    }
    return rtnVal;
}
```

```
    {
        WCHAR * langMultiStrPtr = langMultiStr;
        WCHAR * last = langStr + (langStr[0] == 0xFEFF ? 1 : 0);
        WCHAR * context = last;
        WCHAR * next = wcstok_s(last,L",; :",&context);
        while(next && rtnVal)
        {
            // make sure you validate the user input
            if(SUCCEEDED(StringCchLengthW(last,LOCALE_NAME_MAX_LENGTH,&strLen)) &&
                IsValidLocaleName(next))
            {
                langMultiStrPtr[0] = L'\0';
                rtnVal &= SUCCEEDED(StringCchCatW(langMultiStrPtr,(langMultiStrSize - (langMultiStrPtr - l
                langMultiStrPtr += strLen + 1;
            }
            next = wcstok_s(NULL,L",; :",&context);
            if(next)
                last = next;
        }
        if(rtnVal && (langMultiStrSize - (langMultiStrPtr - langMultiStr))) // make sure there is a double
        {
            langMultiStrPtr[0] = L'\0';
        }
        else // fail and guard anyone whom might use the multi-string
        {
            langMultiStr[0] = L'\0';
            langMultiStr[1] = L'\0';
        }
    }
    return rtnVal;
}
```

```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource 
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource 
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource 
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

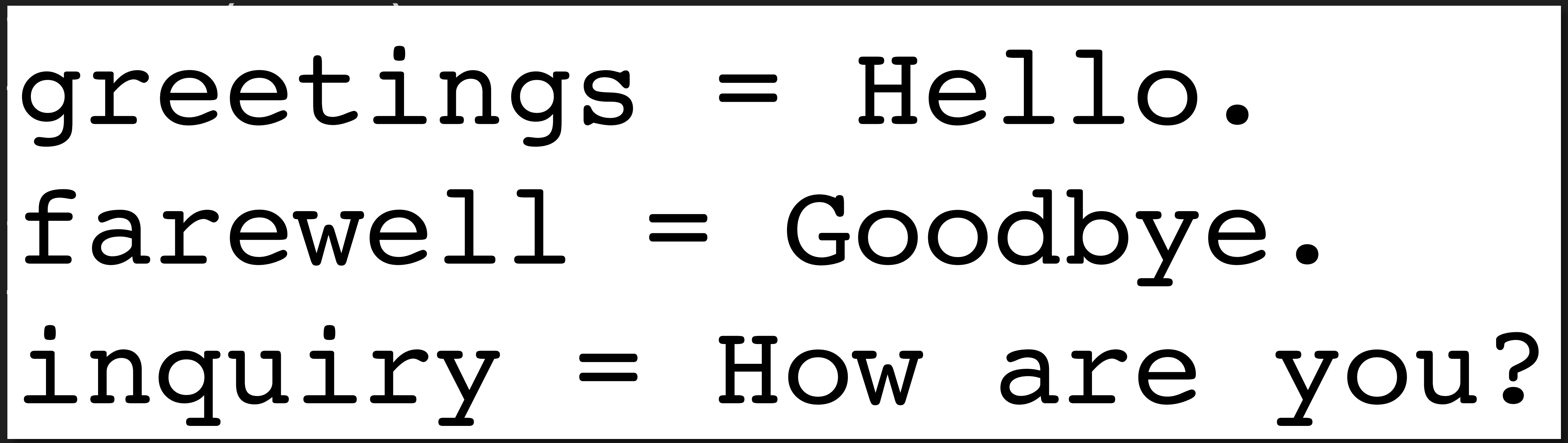
```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

```
    // provides increased security and performance over that of LoadLibrary
    HMODULE resContainer = LoadLibraryExW(HELLO_MODULE_CONTRIVED_FILE_PATH,NULL,LOAD_LIBRARY_AS_IMAGE_RESO
    if(!resContainer)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource co
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }

    // 3. Application parses the resource container to find the appropriate item
    WCHAR szHello[SUFFICIENTLY_LARGE_STRING_BUFFER];
    if(LoadStringW(resContainer,HELLO_MUI_STR_0,szHello,SUFFICIENTLY_LARGE_STRING_BUFFER) == 0)
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to load the resource st
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        FreeLibrary(resContainer);
        return 1; // exit
    }

    // 4. Application presents the discovered resource to the user via UI
    swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"%s MUI",szHello);
    MessageBoxW(NULL,displayBuffer,L"HelloMUI",MB_OK | MB_ICONINFORMATION);

    // 5. Application cleans up memory associated with the resource container after this item is no longer
    if(!FreeLibrary(resContainer))
    {
        swprintf_s(displayBuffer,SUFFICIENTLY_LARGE_ERROR_BUFFER,L"FAILURE: Unable to unload the resource 
        MessageBoxW(NULL,displayBuffer,L"HelloMUI ERROR!",MB_OK | MB_ICONERROR);
        return 1; // exit
    }
```

```java
import java.util.*;

public class I18NSample {

    static public void main(String[] args) {

        String language;
        String country;

        if (args.length != 2) {
            language = new String("en");
            country = new String("US");
        } else {
            language = new String(args[0]);
            country = new String(args[1]);
        }

        Locale currentLocale;
        ResourceBundle messages;

        currentLocale = new Locale(language, country);

        messages = ResourceBundle.getBundle("MessagesBundle", currentLocale);
        System.out.println(messages.getString("greetings"));
        System.out.println(messages.getString("inquiry"));
        System.out.println(messages.getString("farewell"));
    }
}
```

https://docs.oracle.com/javase/tutorial/i18n/intro/after.html

```
import java.util.*;

public class I18NSample {

    static public void main(String[] args) {

        String language;
        String country;

        if (args.length != 2) {
            language = new
            country = new S
        } else {
            language = new
            country = new S
        }

        Locale currentLocale;
        ResourceBundle messages;

        currentLocale = new Locale(language, country);

        messages = ResourceBundle.getBundle("MessagesBundle", currentLocale);
        System.out.println(messages.getString("greetings"));
        System.out.println(messages.getString("inquiry"));
        System.out.println(messages.getString("farewell"));
    }
}
```



```
greetings = Hello.
farewell = Goodbye.
inquiry = How are you?
```

https://doc.qt.io/qt-6/i18n-source-translation.html

```
LoginWidget::LoginWidget()
{
    QLabel *label = new QLabel(tr("Password:"));
    ...
}
```

https://doc.qt.io/qt-6/i18n-source-translation.html

```
LoginWidget::LoginWidget()
{
    QLabel *label = new QLabel(tr("Password:"));
    ...
}
```

```
void FileCopier::showProgress(int done, int total, const QString &currentFile)
{
    label.setText(tr("%1 of %2 files copied.\nCopying: %3").arg(done).arg(total).arg(currentFile));
}
```

https://doc.qt.io/qt-6/i18n-source-translation.html

```
LoginWidget::LoginWidget()
{
    QLabel *label = new QLabel(tr("Password:"));
    ...
}
```

```
void FileCopier::showProgress(int done, int total, const QString &currentFile)
{
    label.setText(tr("%1 of %2 files copied.\nCopying: %3").arg(done).arg(total).arg(currentFile));
}
```

```
MyWindow::MyWindow()
{
    QLabel *senderLabel = new QLabel(tr("Name:"));
    QLabel *recipientLabel = new QLabel(tr("Name:", "recipient"));
    ...
```

https://doc.qt.io/qt-6/i18n-source-translation.html

```
LoginWidget::LoginWidget()
{
    QLabel *label = new QLabel(tr("Password:"));
    ...
}
```

```
void FileCopier::showProgress(int done, int total, const QString &currentFile)
{
    label.setText(tr("%1 of %2 files copied.\nCopying: %3").arg(done).arg(total).arg(currentFile));
}
```

```
MyWindow::MyWindow()
{
    QLabel *senderLabel = new QLabel(tr("Name:"));
    QLabel *recipientLabel = new QLabel(tr("Name:", "recipient"));
    ...
```

```
showMessage(tr("%n message(s) saved", "", messages.count()));
```

https://doc.qt.io/qt-6/i18n-source-translation.html

```
LoginWidget::LoginWidget()
{
    QLabel *label = new QLabel(tr("Password:"));
    ...
}
```

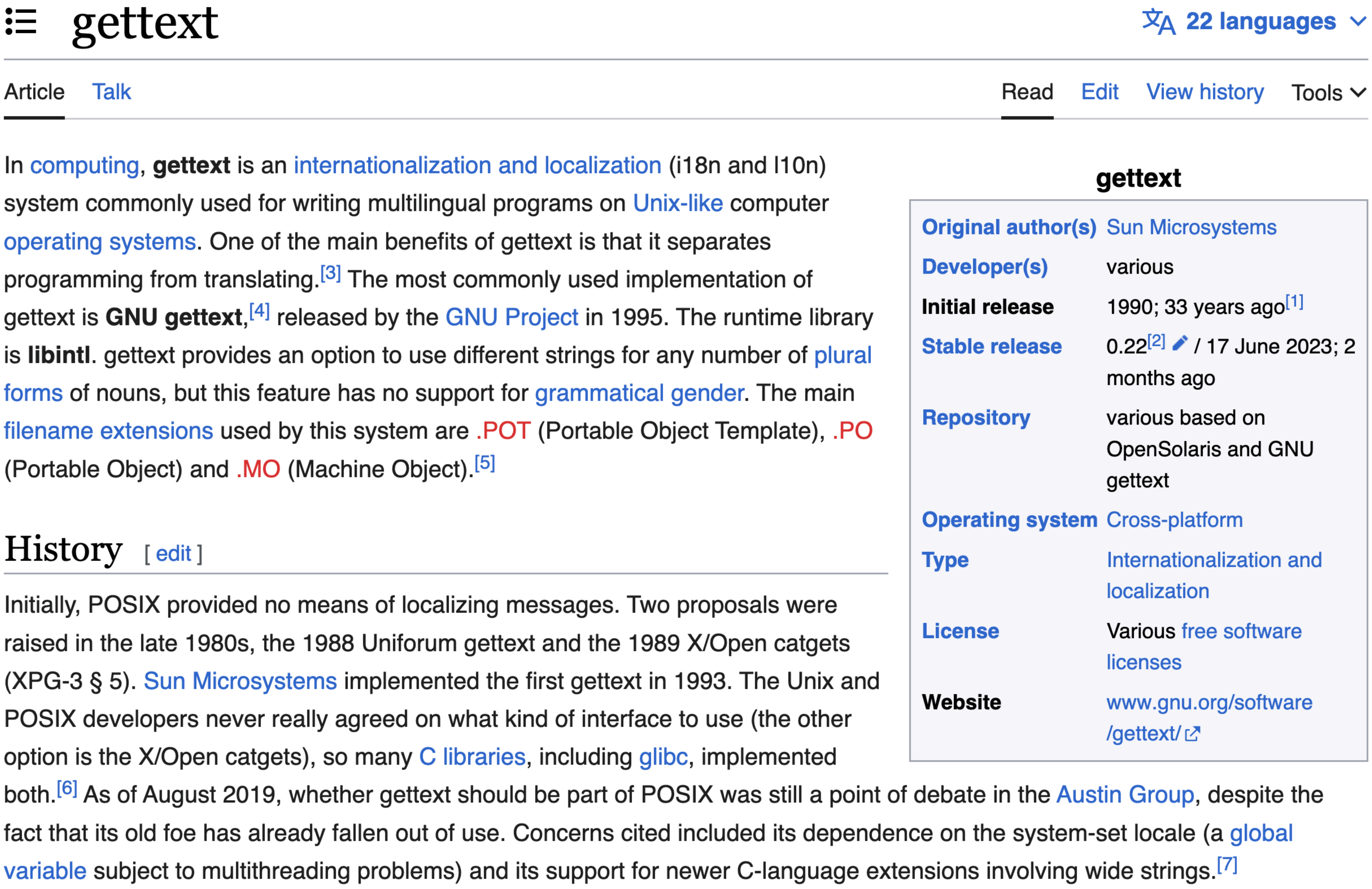
```
void FileCopier::showProgress(int done, int total, const QString &currentFile)
{
    label.setText(tr("%1 of %2 files copied.\nCopying: %3").arg(done).arg(total).arg(currentFile));
}
```

```
MyWindow::MyWindow()
{
    QLabel *senderLabel = new QLabel(tr("Name:"));
    QLabel *recipientLabel = new QLabel(tr("Name:", "recipient"));
    ...
```

```
showMessage(tr("%n message(s) saved", "", messages.count()));
```

```
//: This name refers to a host name.
hostNameLabel->setText(tr("Name:"));
```

https://doc.qt.io/qt-6/i18n-source-translation.html



gettext

In computing, **gettext** is an internationalization and localization (i18n and l10n) system commonly used for writing multilingual programs on Unix-like computer operating systems. One of the main benefits of gettext is that it separates programming from translating.[3] The most commonly used implementation of gettext is **GNU gettext**,[4] released by the GNU Project in 1995. The runtime library is **libintl**. gettext provides an option to use different strings for any number of plural forms of nouns, but this feature has no support for grammatical gender. The main filename extensions used by this system are .POT (Portable Object Template), .PO (Portable Object) and .MO (Machine Object).[5]

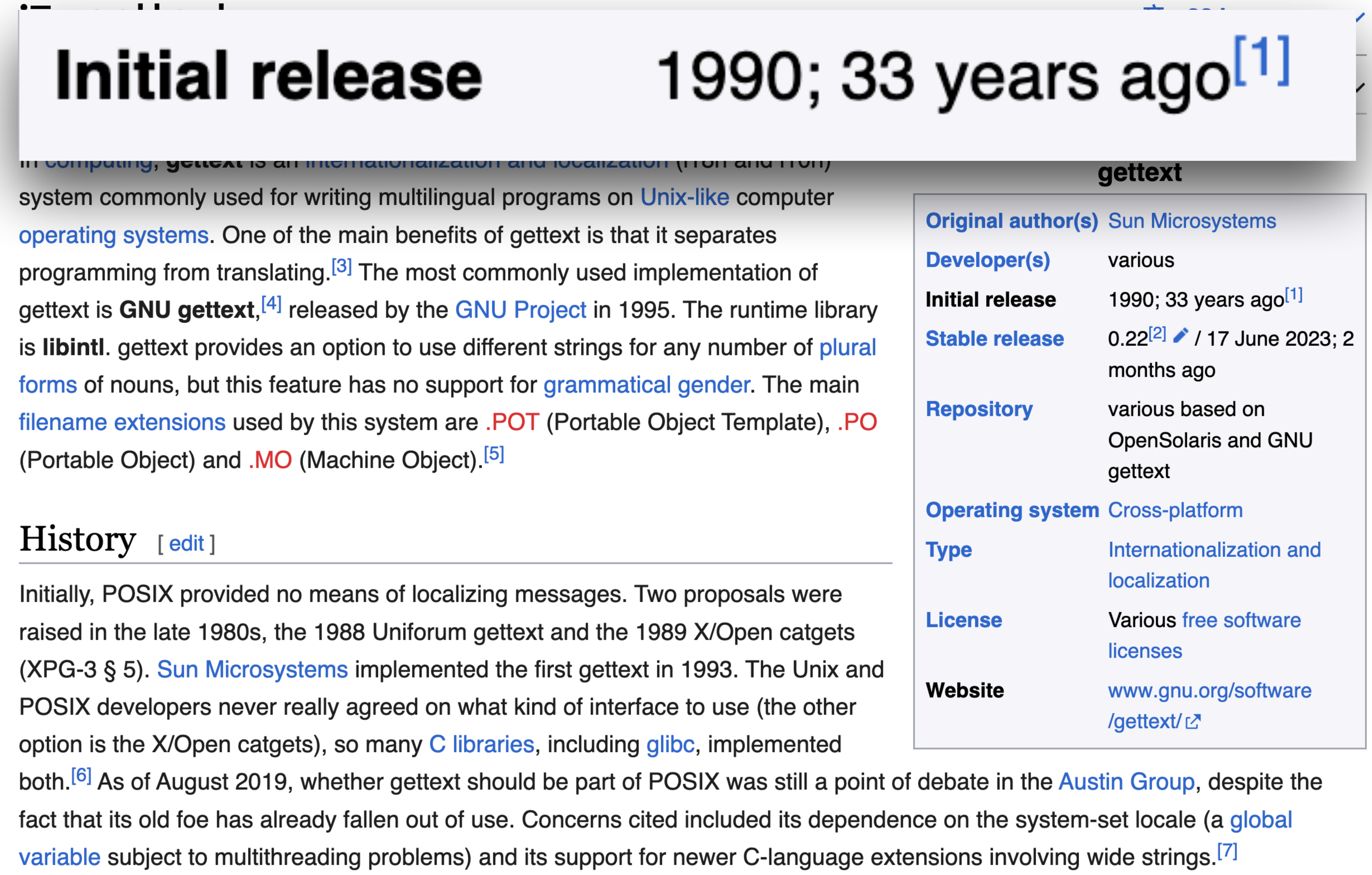
gettext

Original author(s)	Sun Microsystems
Developer(s)	various
Initial release	1990; 33 years ago[1]
Stable release	0.22[2] / 17 June 2023; 2 months ago
Repository	various based on OpenSolaris and GNU gettext
Operating system	Cross-platform
Type	Internationalization and localization
License	Various free software licenses
Website	www.gnu.org/software/gettext/

History [edit]

Initially, POSIX provided no means of localizing messages. Two proposals were raised in the late 1980s, the 1988 Uniforum gettext and the 1989 X/Open catgets (XPG-3 § 5). Sun Microsystems implemented the first gettext in 1993. The Unix and POSIX developers never really agreed on what kind of interface to use (the other option is the X/Open catgets), so many C libraries, including glibc, implemented both.[6] As of August 2019, whether gettext should be part of POSIX was still a point of debate in the Austin Group, despite the fact that its old foe has already fallen out of use. Concerns cited included its dependence on the system-set locale (a global variable subject to multithreading problems) and its support for newer C-language extensions involving wide strings.[7]

https://wikipedia.org/wiki/Gettext



Initial release 1990; 33 years ago[1]

In computing, **gettext** is an internationalization and localization (i18n and l10n) system commonly used for writing multilingual programs on Unix-like computer operating systems. One of the main benefits of gettext is that it separates programming from translating.[3] The most commonly used implementation of gettext is **GNU gettext**,[4] released by the GNU Project in 1995. The runtime library is **libintl**. gettext provides an option to use different strings for any number of plural forms of nouns, but this feature has no support for grammatical gender. The main filename extensions used by this system are .POT (Portable Object Template), .PO (Portable Object) and .MO (Machine Object).[5]

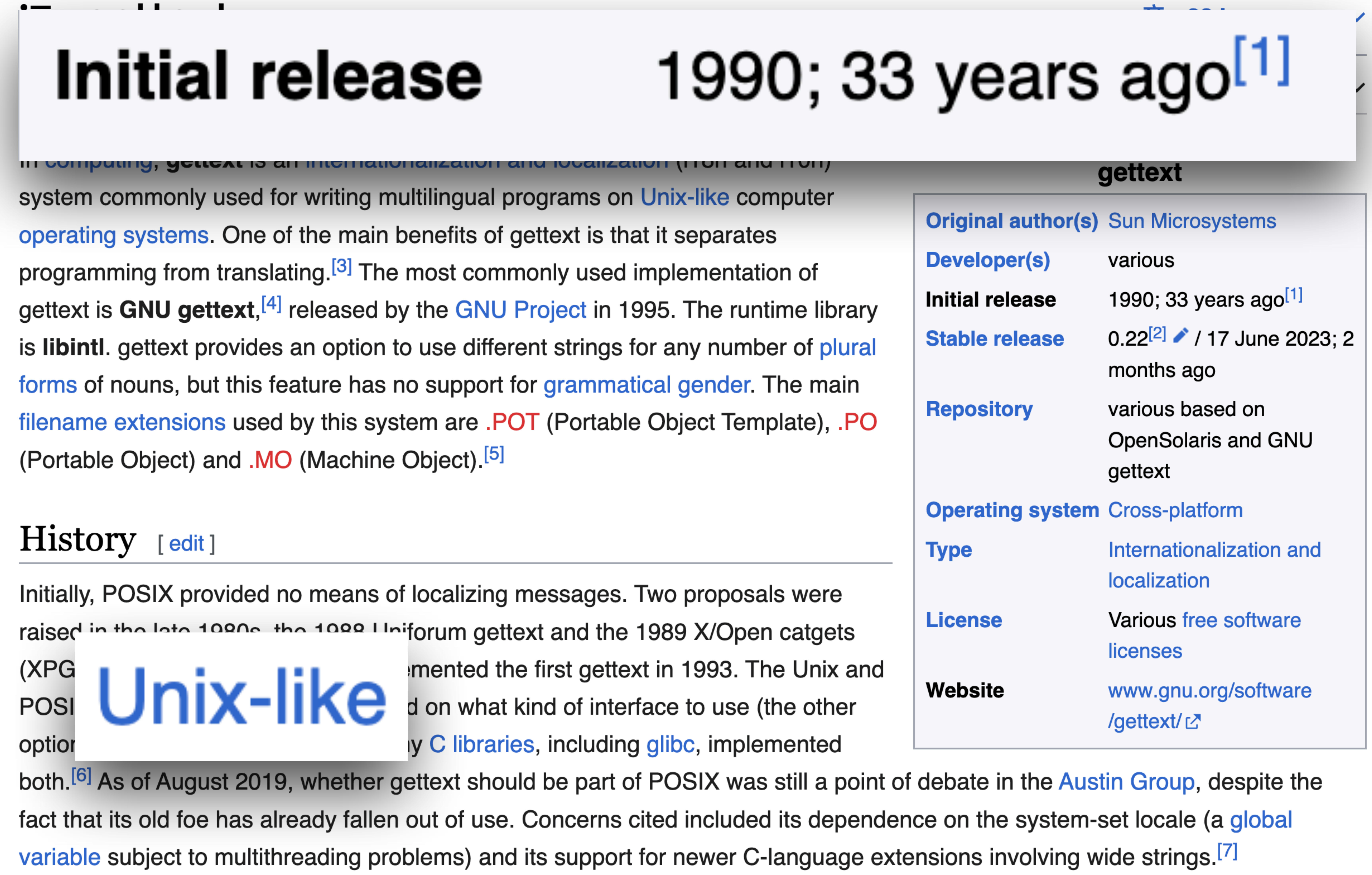
gettext

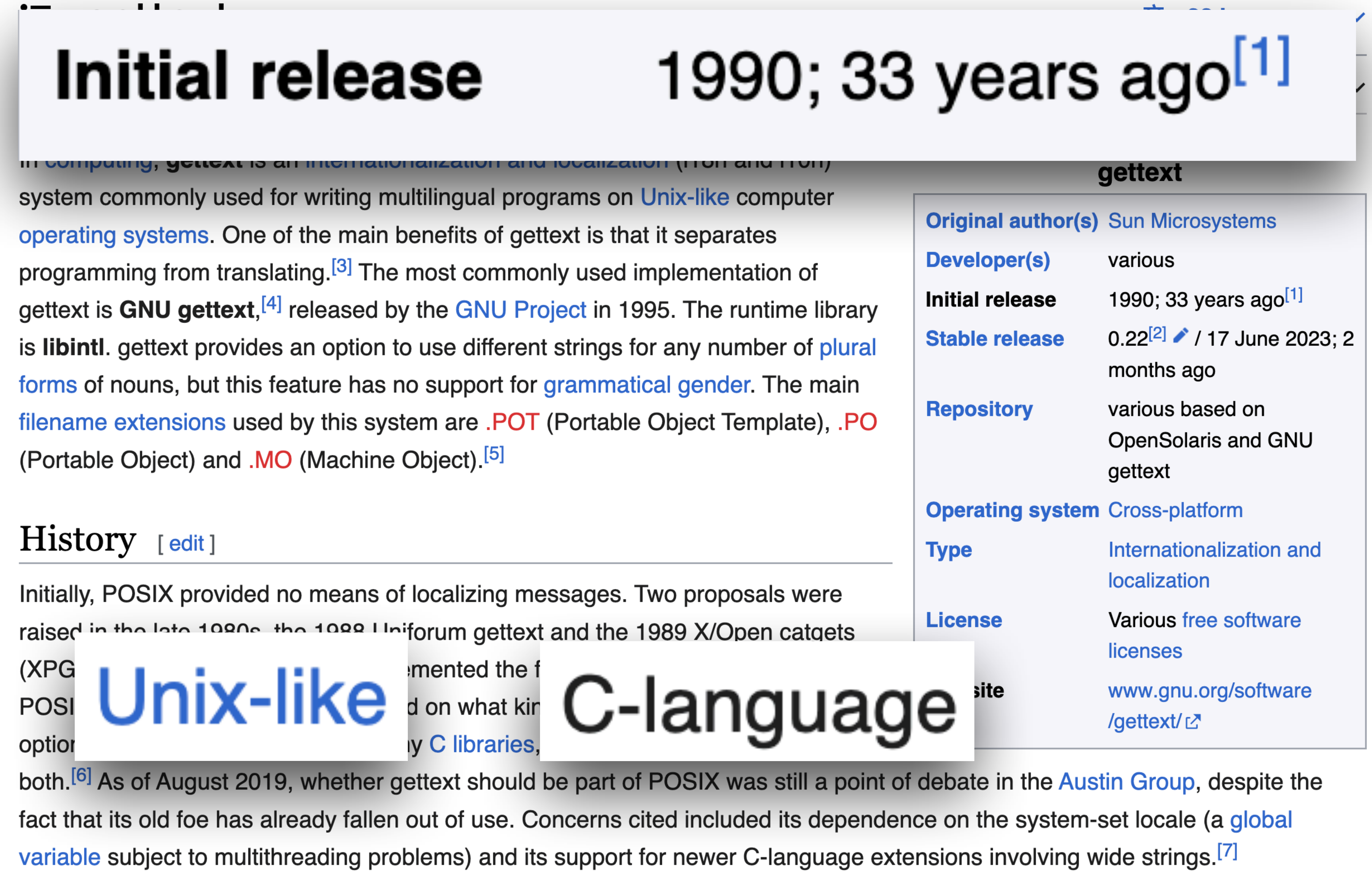
Original author(s)	Sun Microsystems
Developer(s)	various
Initial release	1990; 33 years ago[1]
Stable release	0.22[2] ✎ / 17 June 2023; 2 months ago
Repository	various based on OpenSolaris and GNU gettext
Operating system	Cross-platform
Type	Internationalization and localization
License	Various free software licenses
Website	www.gnu.org/software/gettext/

History [edit]

Initially, POSIX provided no means of localizing messages. Two proposals were raised in the late 1980s, the 1988 Uniforum gettext and the 1989 X/Open catgets (XPG-3 § 5). Sun Microsystems implemented the first gettext in 1993. The Unix and POSIX developers never really agreed on what kind of interface to use (the other option is the X/Open catgets), so many C libraries, including glibc, implemented both.[6] As of August 2019, whether gettext should be part of POSIX was still a point of debate in the Austin Group, despite the fact that its old foe has already fallen out of use. Concerns cited included its dependence on the system-set locale (a global variable subject to multithreading problems) and its support for newer C-language extensions involving wide strings.[7]

https://wikipedia.org/wiki/Gettext





Initial release 1990; 33 years ago[1]

In computing, **gettext** is an internationalization and localization (i18n and l10n) system commonly used for writing multilingual programs on Unix-like computer operating systems. One of the main benefits of gettext is that it separates programming from translating.[3] The most commonly used implementation of gettext is **GNU gettext**,[4] released by the GNU Project in 1995. The runtime library is **libintl**. gettext provides an option to use different strings for any number of plural forms of nouns, but this feature has no support for grammatical gender. The main filename extensions used by this system are .POT (Portable Object Template), .PO (Portable Object) and .MO (Machine Object).[5]

History [edit]

Initially, POSIX provided no means of localizing messages. Two proposals were raised in the late 1980s, the 1988 Uniforum gettext and the 1989 X/Open catgets (XPG mented the f POSI d on what kir optio ly C libraries, both.[6] As of August 2019, whether gettext should be part of POSIX was still a point of debate in the Austin Group, despite the fact that its old foe has already fallen out of use. Concerns cited included its dependence on the system-set locale (a global variable subject to multithreading problems) and its support for newer C-language extensions involving wide strings.[7]

gettext

Original author(s)	Sun Microsystems
Developer(s)	various
Initial release	1990; 33 years ago[1]
Stable release	0.22[2] / 17 June 2023; 2 months ago
Repository	various based on OpenSolaris and GNU gettext
Operating system	Cross-platform
Type	Internationalization and localization
License	Various free software licenses
site	www.gnu.org/software/gettext/

Unix-like

C-language

https://wikipedia.org/wiki/Gettext

Initial release 1990; 33 years ago[1]

In computing, gettext is an internationalization and localization (i18n and l10n) system commonly used for writing multilingual programs on Unix-like computer operating systems. One of the main benefits of gettext is that it separates programming from translating.[3] The most commonly used implementation of gettext is **GNU gettext**,[4] released by the GNU Project in 1995. The runtime library

gettext

Original author(s)	Sun Microsystems
Developer(s)	various
Initial release	1990; 33 years ago[1]
Operating system	Cross-platform
Type	Internationalization and localization
License	Various free software licenses
...site	www.gnu.org/software/gettext/

GNU gettext,[4] released by the GNU Project in 1995.

wiki/Gettext

History [edit]

Initially, POSIX provided no means of localizing messages. Two proposals were raised in the late 1980s, the 1988 Uniforum gettext and the 1989 X/Open catgets (XPG ...mented the f... POSI... d on what kin... option... y C libraries, ... both.[6] As of August 2019, whether gettext should be part of POSIX was still a point of debate in the Austin Group, despite the fact that its old foe has already fallen out of use. Concerns cited included its dependence on the system-set locale (a global variable subject to multithreading problems) and its support for newer C-language extensions involving wide strings.[7]

Unix-like

C-language

Initial release 1990; 33 years ago[1]

In computing, gettext is an internationalization and localization (i18n and l10n) system commonly used for writing multilingual programs on Unix-like computer operating systems. One of the main benefits of gettext is that it separates programming from translating.[3] The most commonly used implementation of gettext is **GNU gettext**,[4] released by the GNU Project in 1995. The runtime library

gettext

Original author(s)	Sun Microsystems
Developer(s)	various
Initial release	1990; 33 years ago[1]
Operating system	Cross-platform
Type	Internationalization and localization
License	Various free software licenses
site	www.gnu.org/software/gettext/

GNU gettext,[4] released by the GNU Project in 1995.

wiki/Gettext

History [edit]

Initially, POSIX provided no means of localizing messages. Two proposals were raised in the late 1980s, the 1988 Uniforum gettext and the 1989 X/Open catgets (XPG mented the f POSI d on what kin option y C libraries, both.[6] As of August 2019, whether gettext should be part of POSIX was still a point of debate in the Austin Group, despite the fact that its old foe has already fallen out of use. Concerns cited included its dependence on the system-set locale (a global variable subject to multithreading problems) and its support for newer C-language extensions involving wide strings.[7]

~~Unix-like~~ Multi

C-language

Initial release 1990; 33 years ago[1]

In computing, **gettext** is an internationalization and localization (i18n and l10n) system commonly used for writing multilingual programs on Unix-like computer operating systems. One of the main benefits of gettext is that it separates programming from translating.[3] The most commonly used implementation of gettext is **GNU gettext**,[4] released by the GNU Project in 1995. The runtime library

GNU gettext,[4] released by the GNU Project in 1995.

gettext	
Original author(s)	Sun Microsystems
Developer(s)	various
Initial release	1990; 33 years ago[1]
Operating system	Cross-platform
Type	Internationalization and localization
License	Various free software licenses
Website	www.gnu.org/software/gettext/

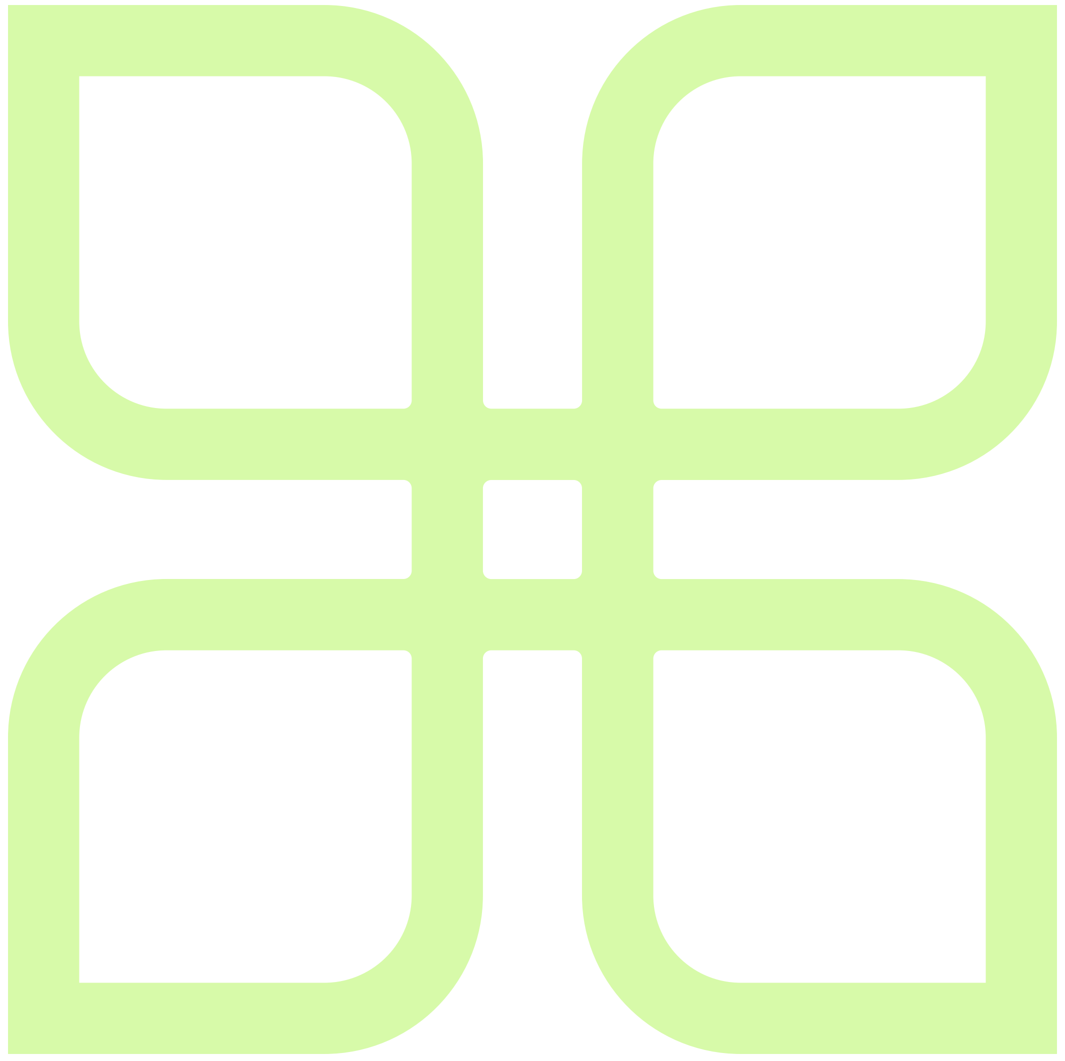
History [edit]

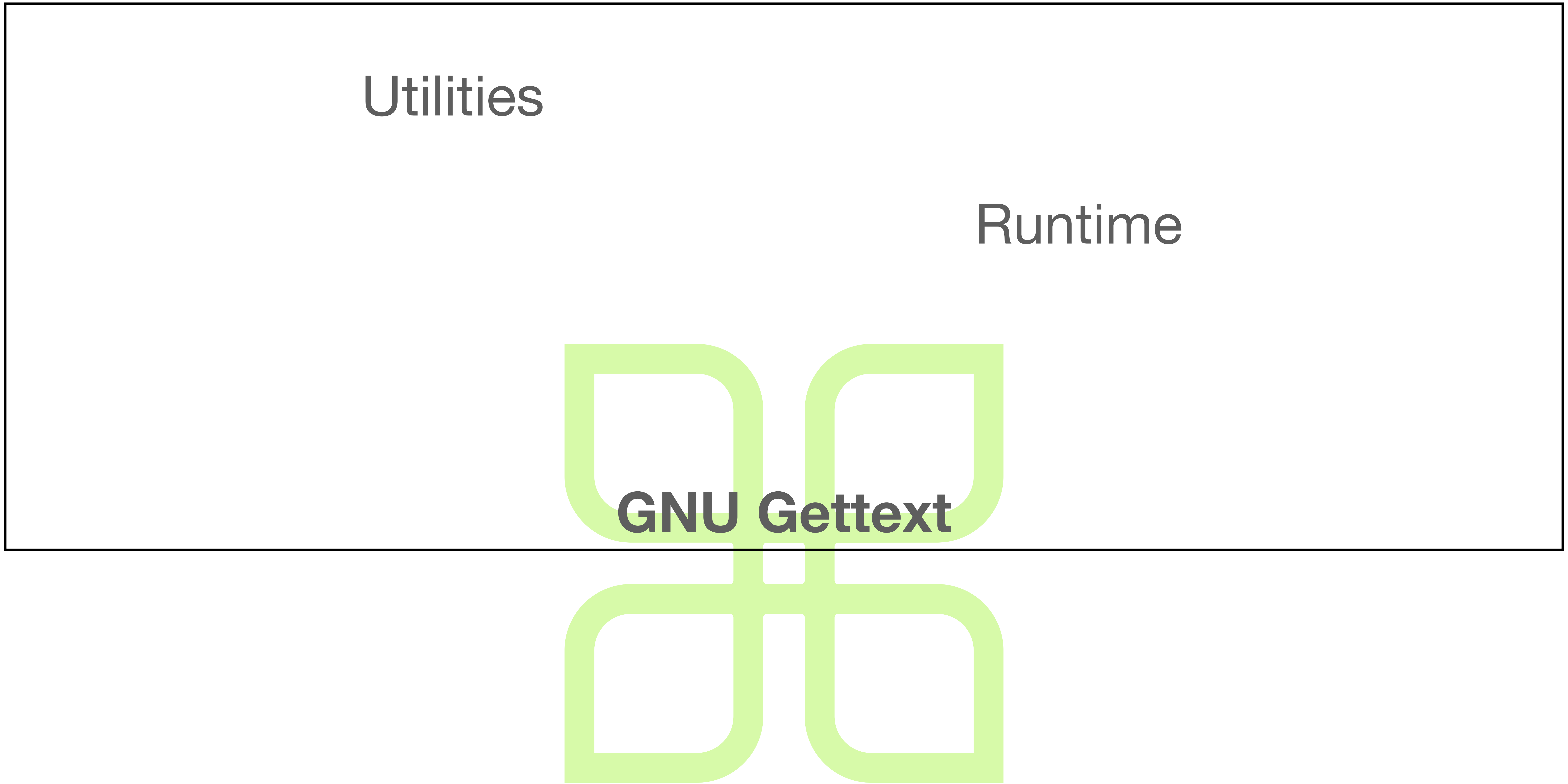
Initially, POSIX provided no means of localizing messages. Two proposals were raised in the late 1980s, the 1988 Uniforum gettext and the 1989 X/Open catgets (XPG … mented the f … POSI … d … what kin … option … both.[6] As of August 2019, whether gettext should be part of POSIX was still a point of debate in the Austin Group, despite the fact that its old foe has already fallen out of use. Concerns cited included its dependence on the system-set locale (a global variable subject to multithreading problems) and its support for newer C-language extensions involving wide strings.[7]

~~Unix-like~~ Multi

~~C-language~~ Multi

wiki/Gettext

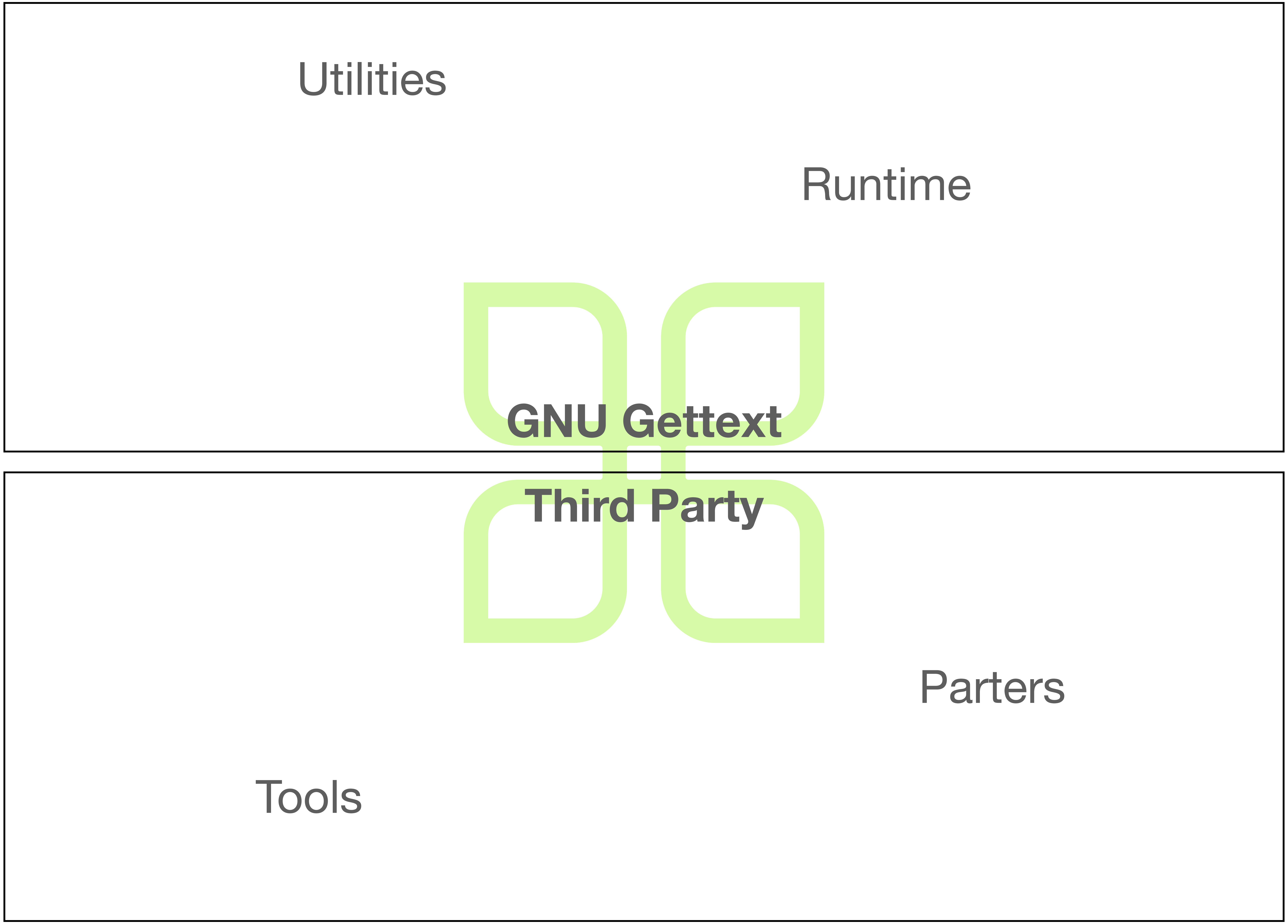




Utilities

Runtime

GNU Gettext



Utilities

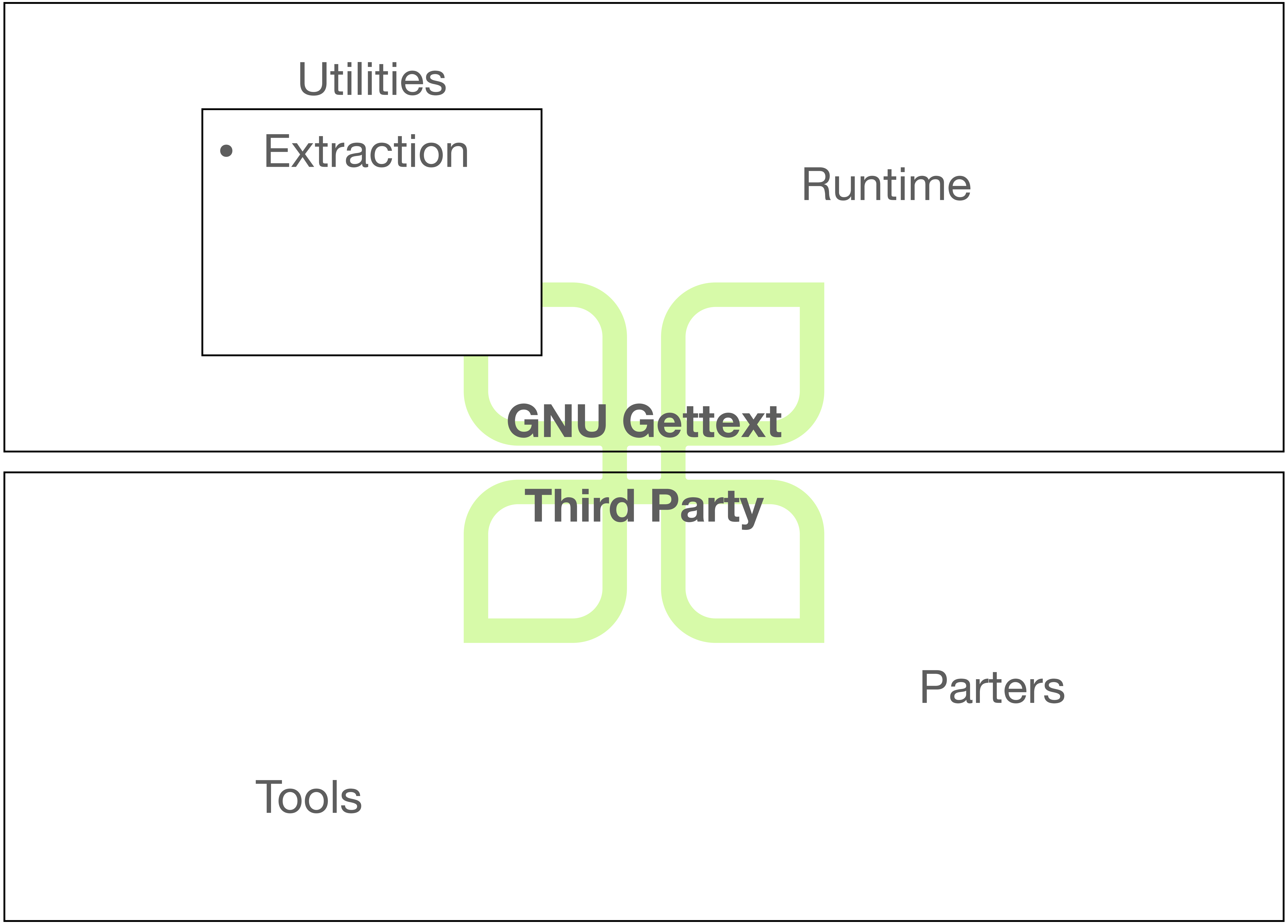
Runtime

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction

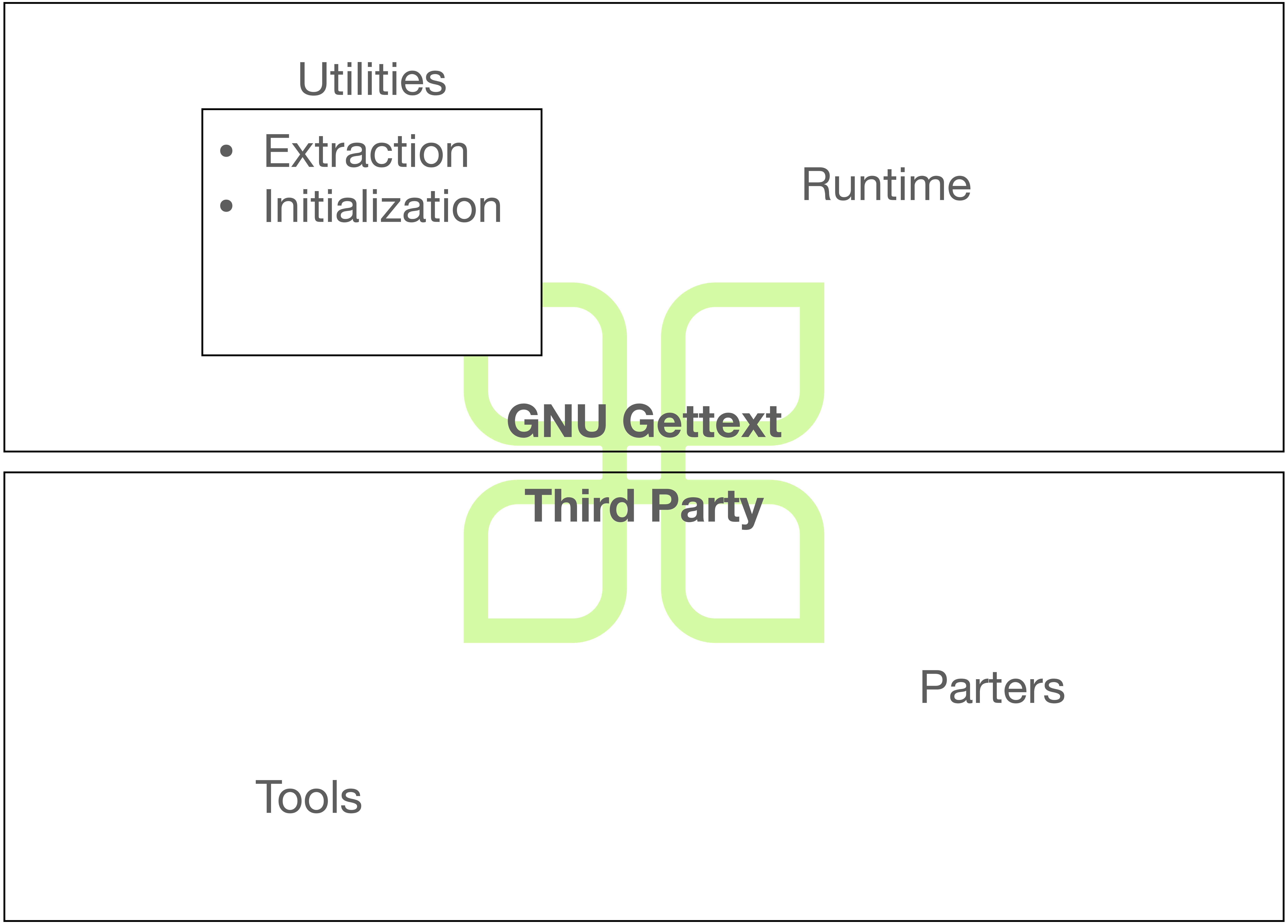
Runtime

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction
- Initialization

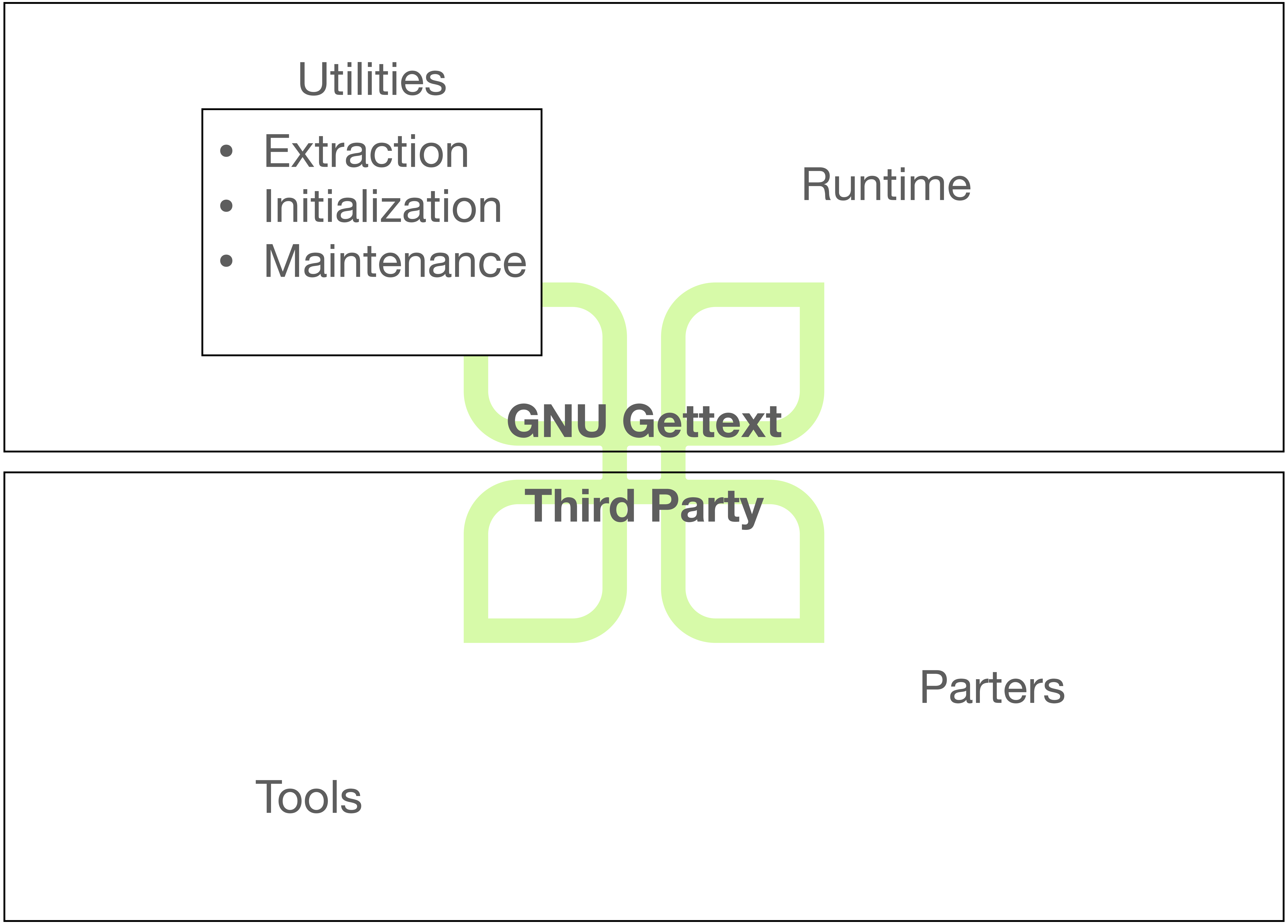
Runtime

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction
- Initialization
- Maintenance

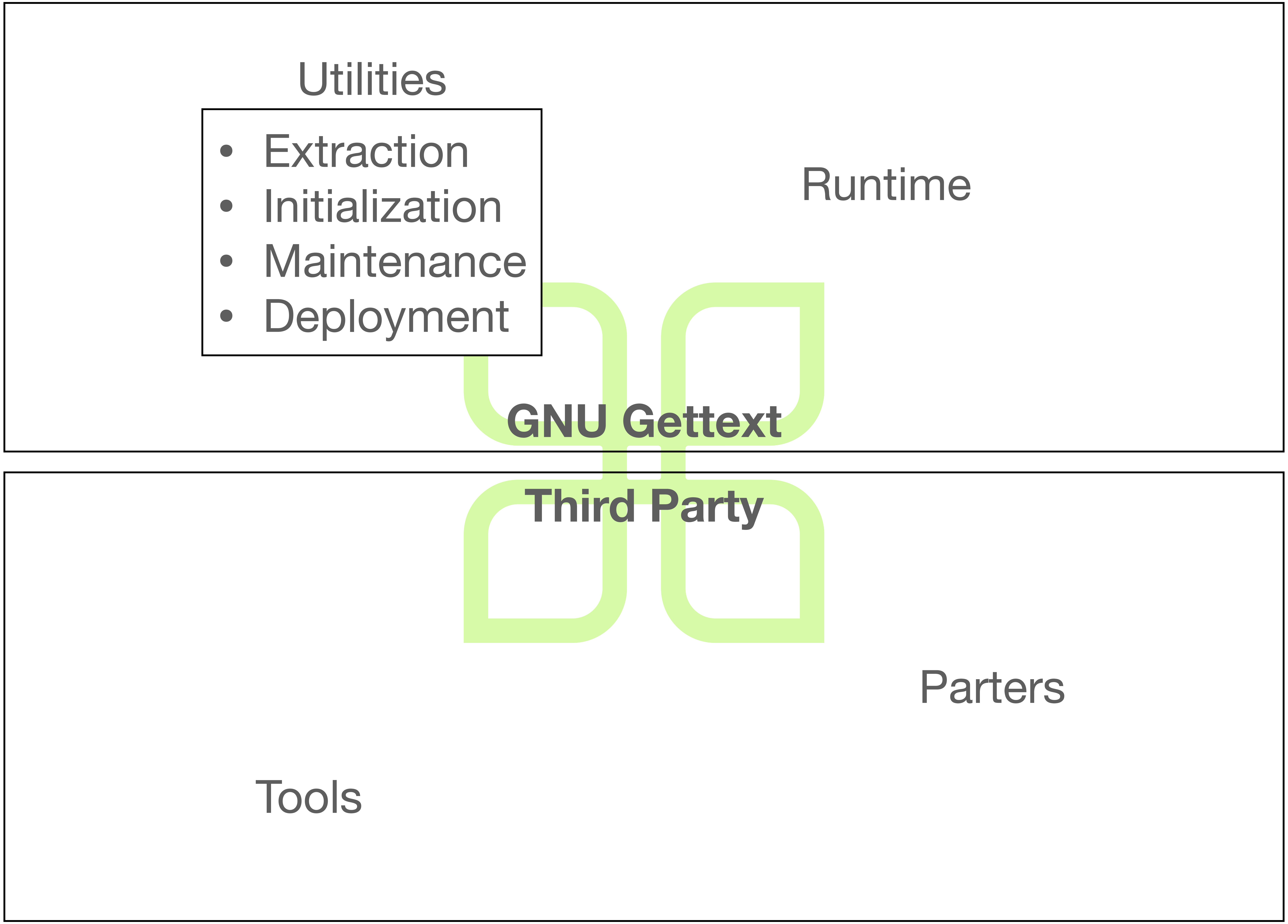
Runtime

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

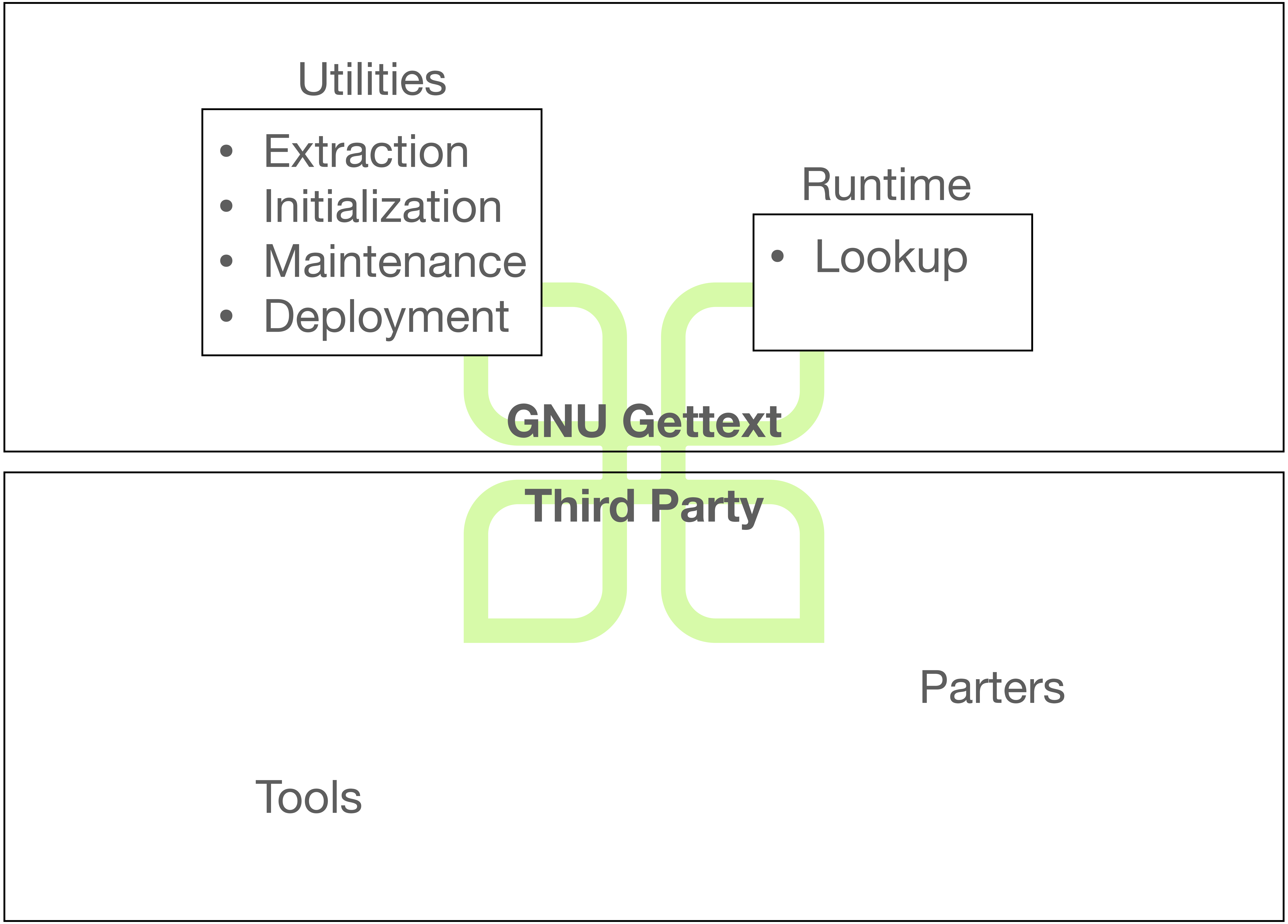
Runtime

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

Runtime

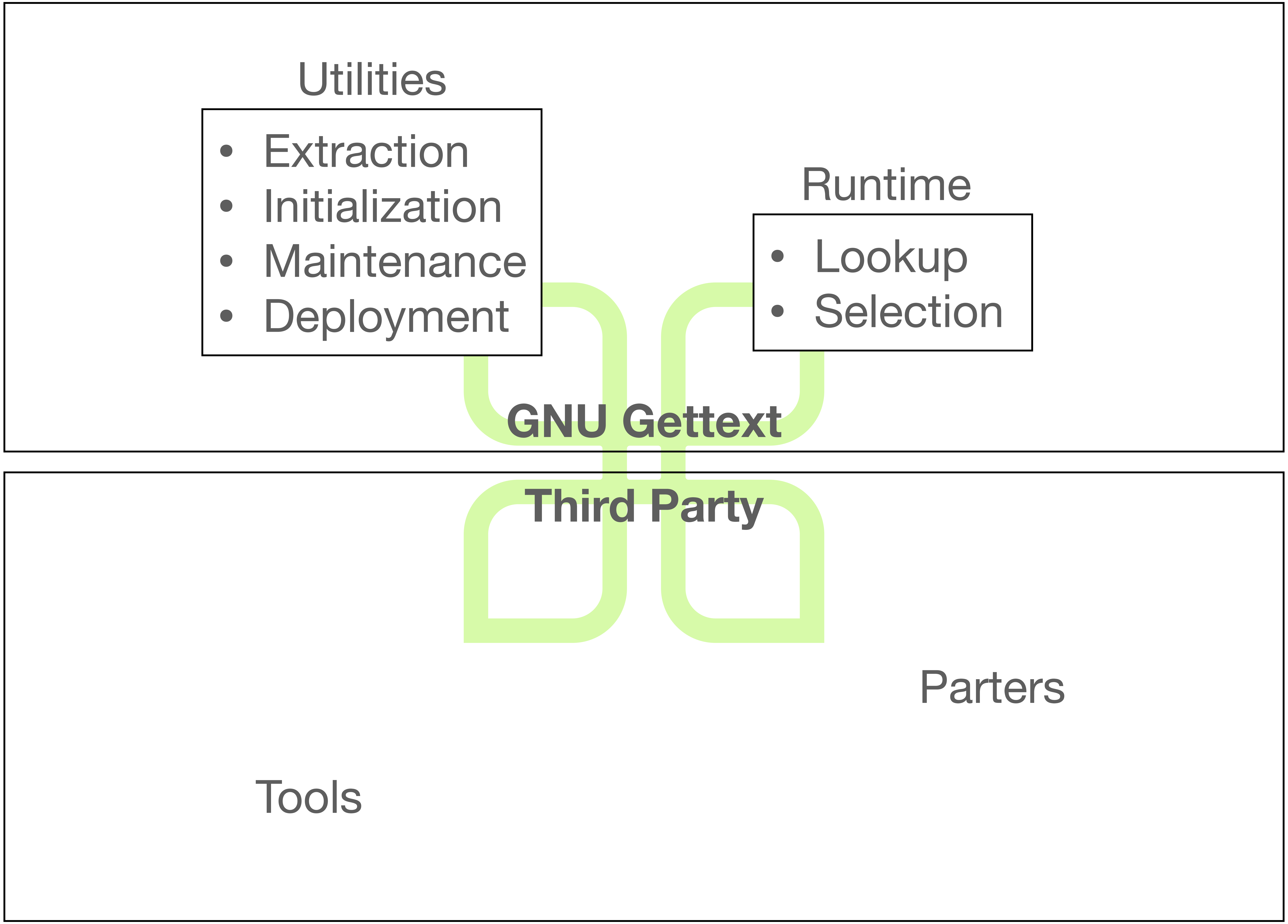
- Lookup

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

Runtime

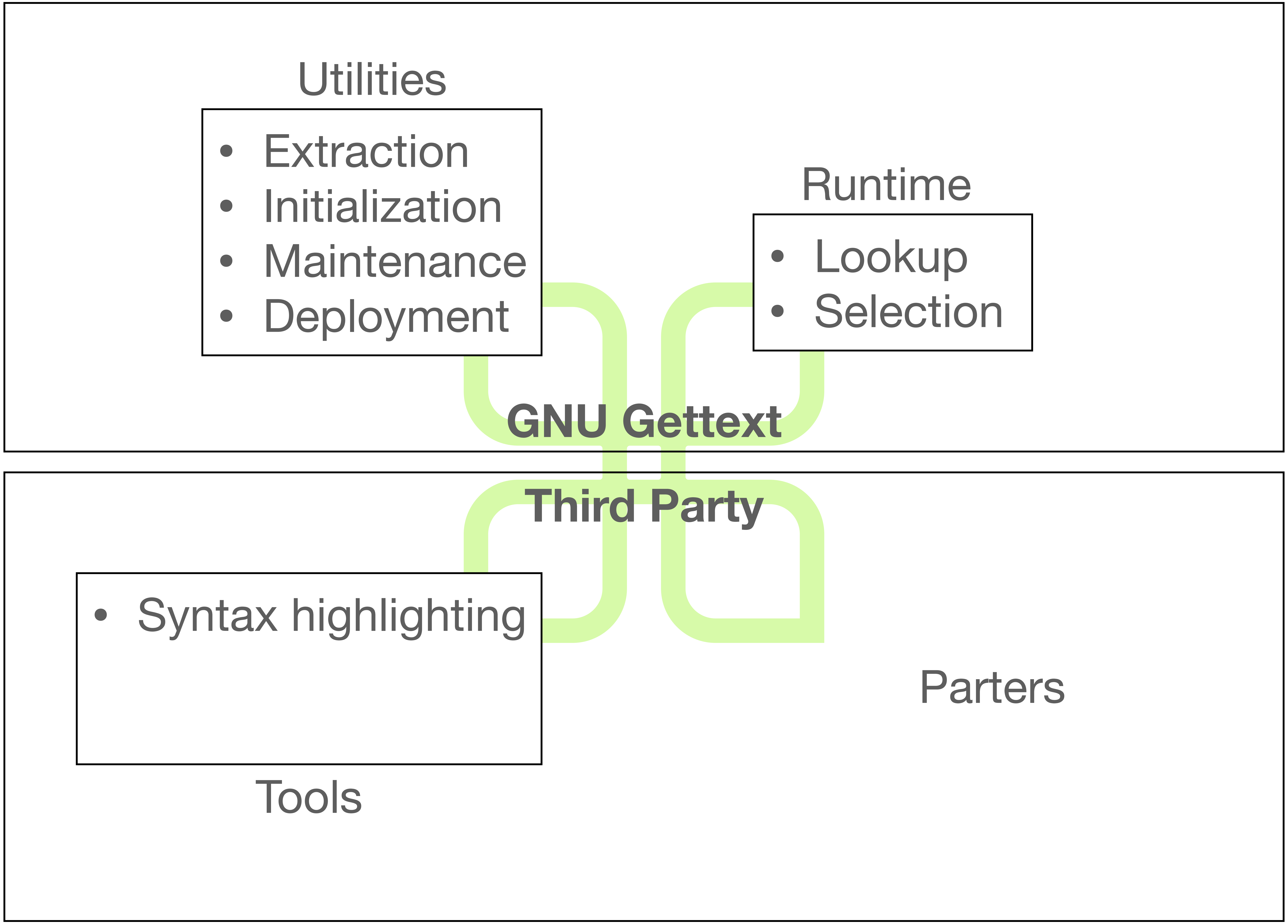
- Lookup
- Selection

GNU Gettext

Third Party

Parters

Tools



Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

Runtime

- Lookup
- Selection

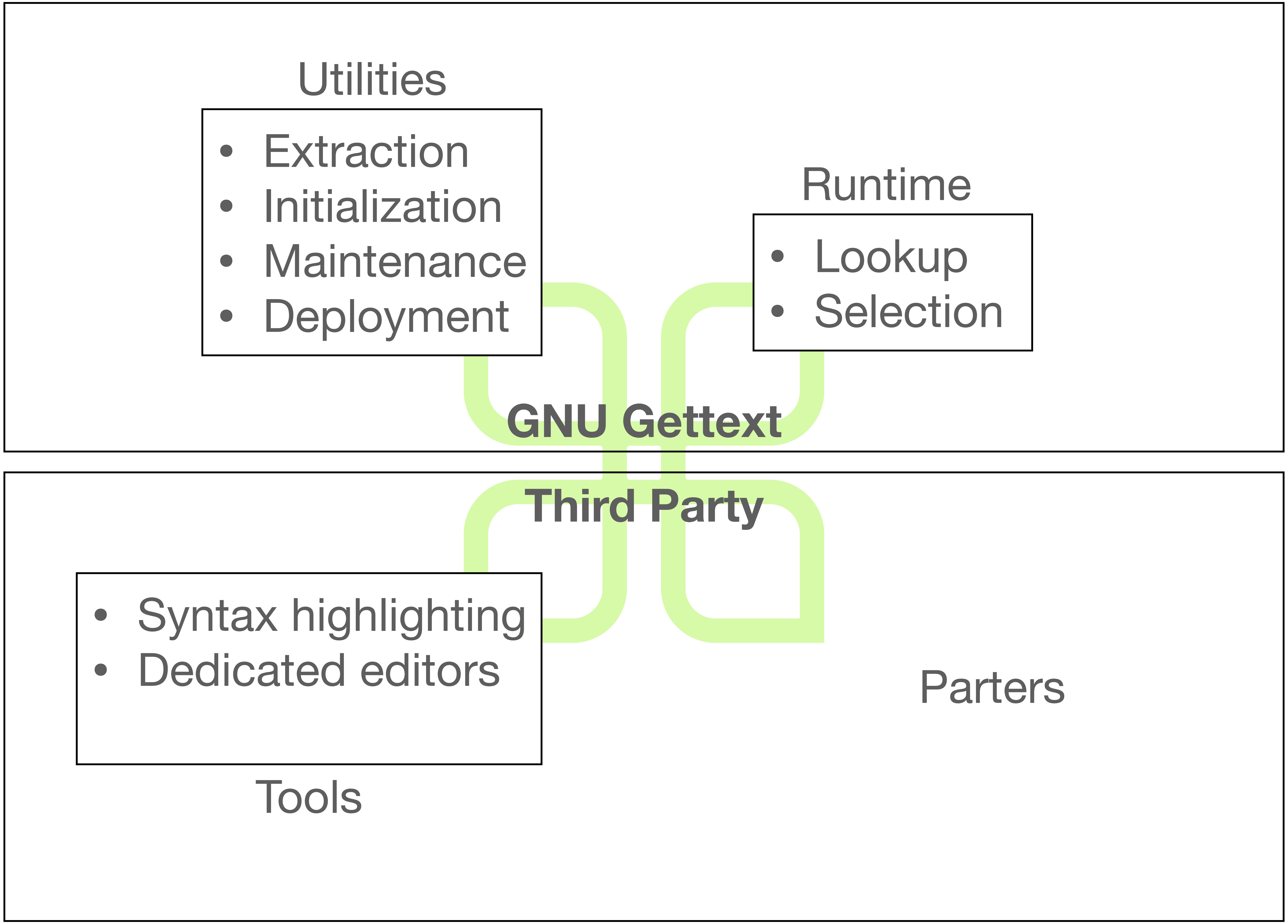
GNU Gettext

Third Party

- Syntax highlighting

Tools

Parters



GNU Gettext

Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

Runtime

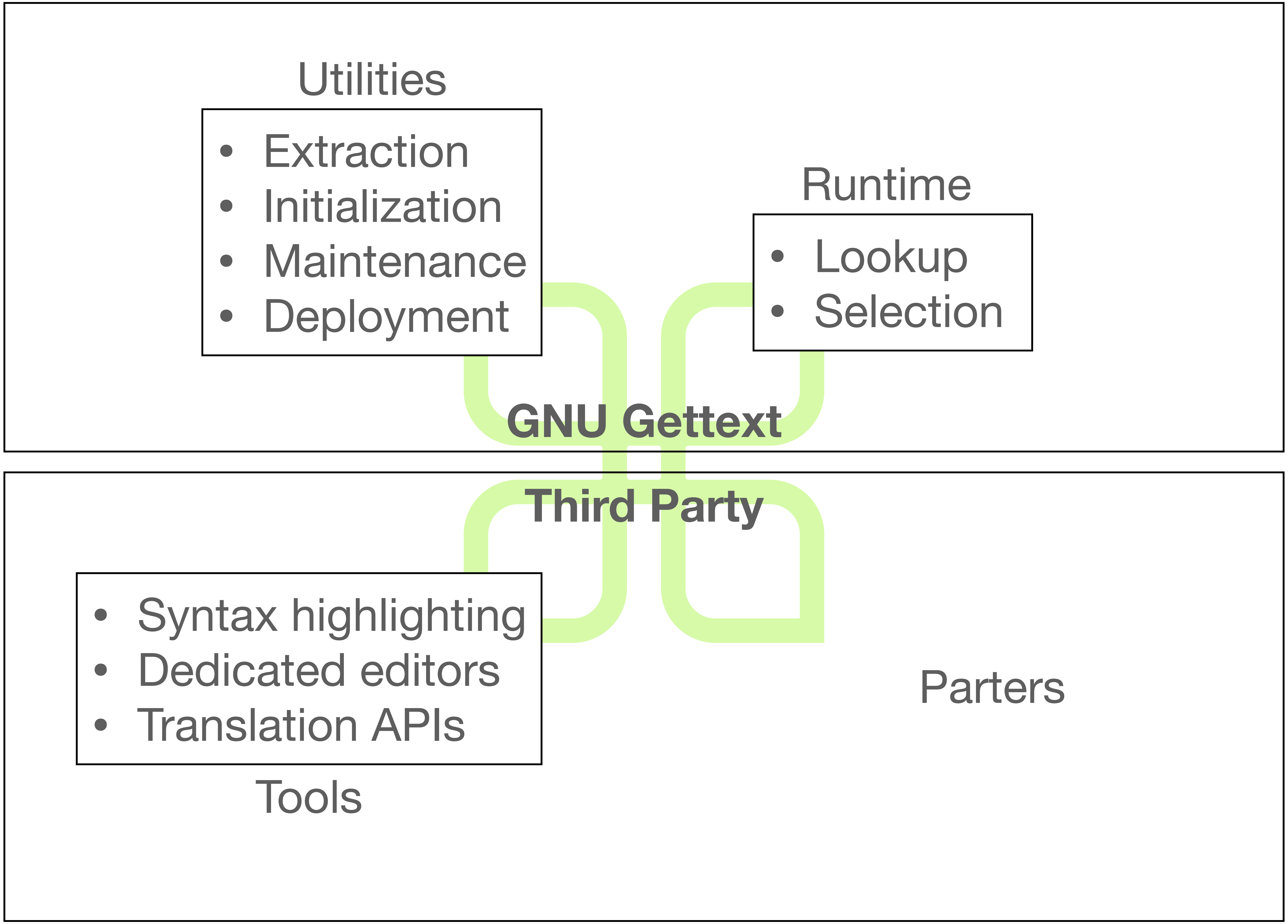
- Lookup
- Selection

Third Party

- Syntax highlighting
- Dedicated editors

Tools

Parters



Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

Runtime

- Lookup
- Selection

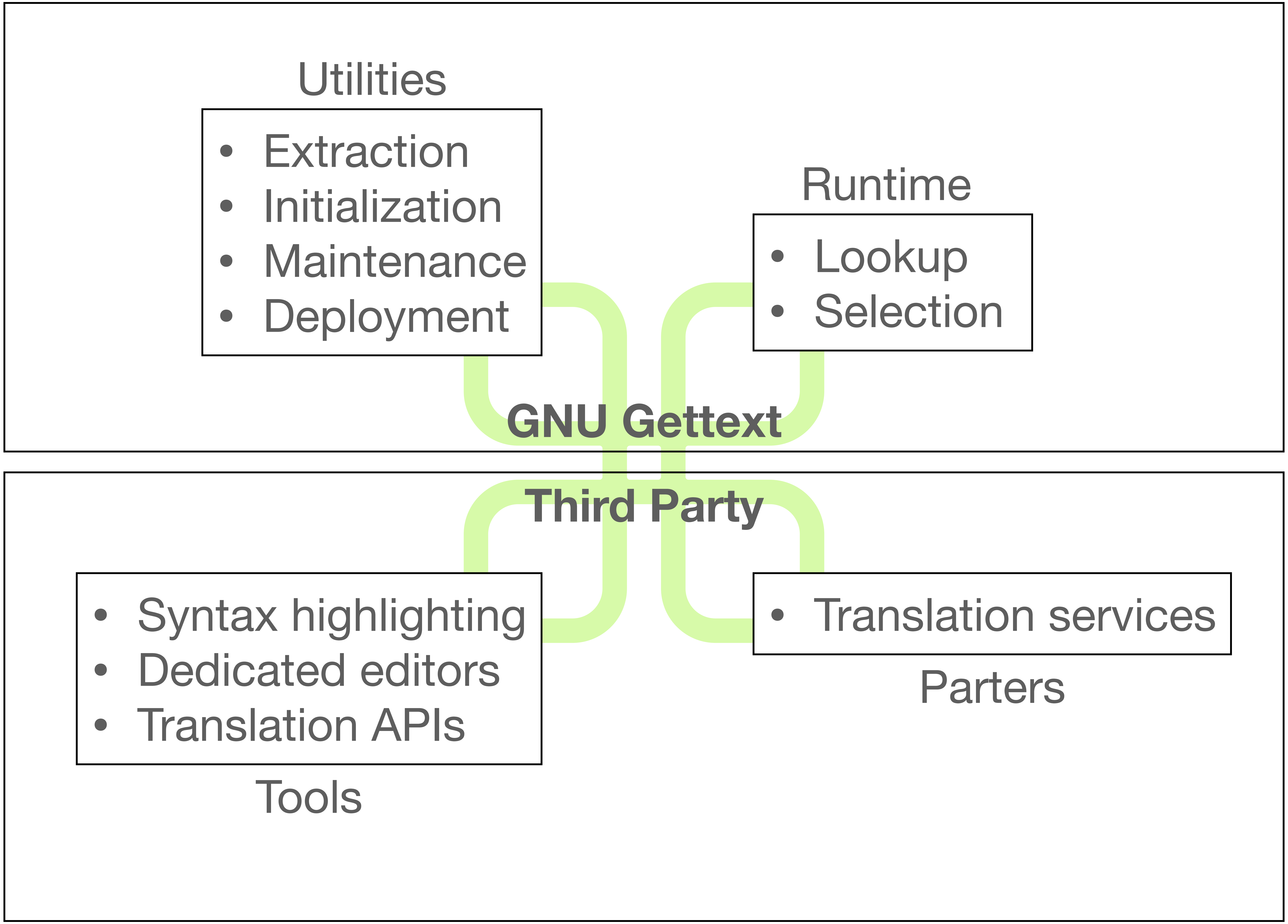
GNU Gettext

Third Party

- Syntax highlighting
- Dedicated editors
- Translation APIs

Tools

Parters



Utilities

- Extraction
- Initialization
- Maintenance
- Deployment

Runtime

- Lookup
- Selection

GNU Gettext

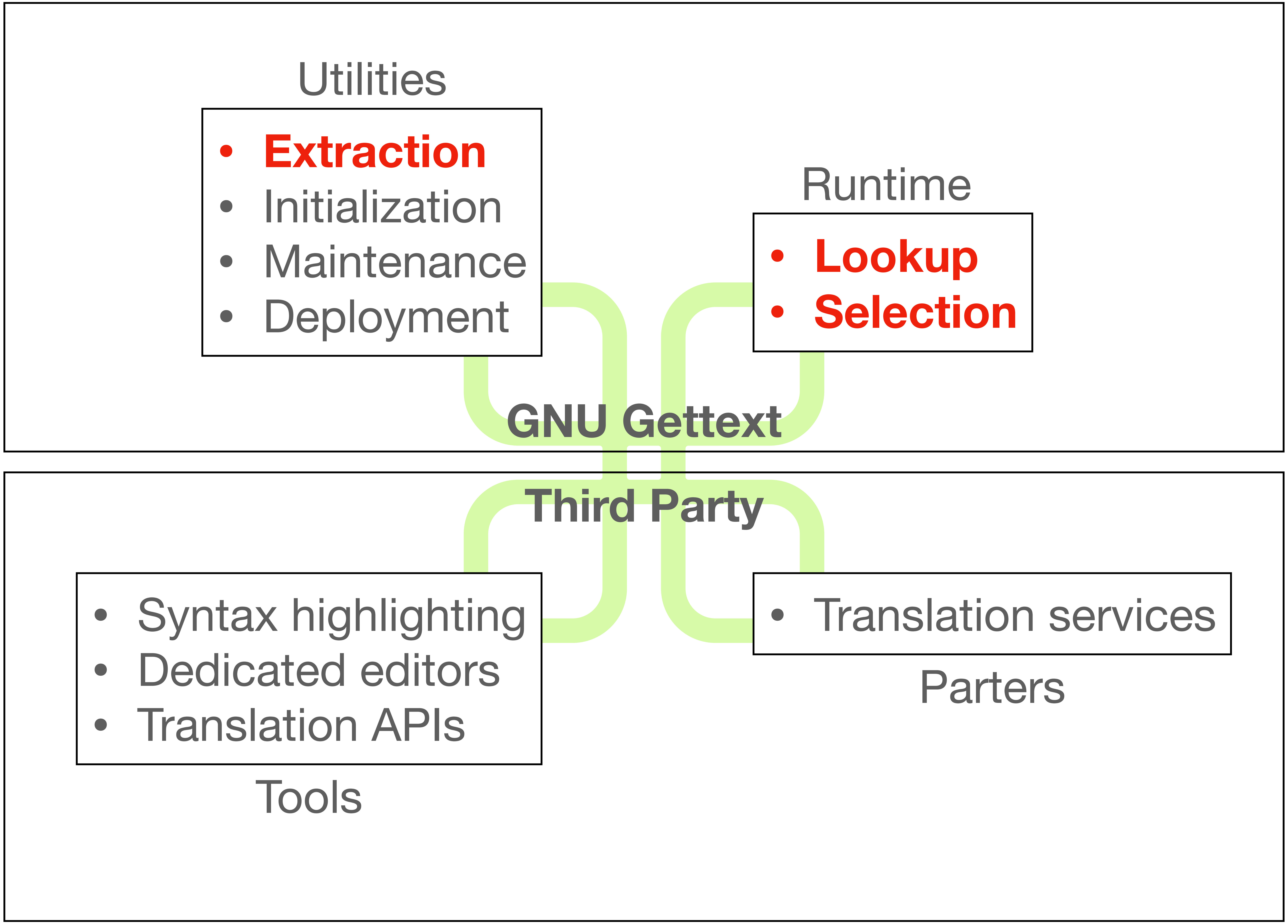
Third Party

- Syntax highlighting
- Dedicated editors
- Translation APIs

Tools

- Translation services

Parters



Utilities

- **Extraction**
- Initialization
- Maintenance
- Deployment

Runtime

- **Lookup**
- **Selection**

GNU Gettext

Third Party

- Syntax highlighting
- Dedicated editors
- Translation APIs

Tools

- Translation services

Parters

https://www.gnu.org/software/gettext/manual/

```
#include <libintl.h>

puts(gettext(“This is a translatable string.\n”));
printf(gettext("String \"%s\" has %d characters.\n”),
       s, strlen(s));
```

https://
www.gnu.org/
software/
gettext/manual/

```
#include <libintl.h>

puts(gettext(“This is a translatable string.\n”));
printf(gettext("String \"%s\" has %d characters.\n”),
       s, strlen(s));
// TRANSLATORS: Comment regarding following string.
printf(pgettext("Message recipient", "Name: %s”),
       recipient_name);
```

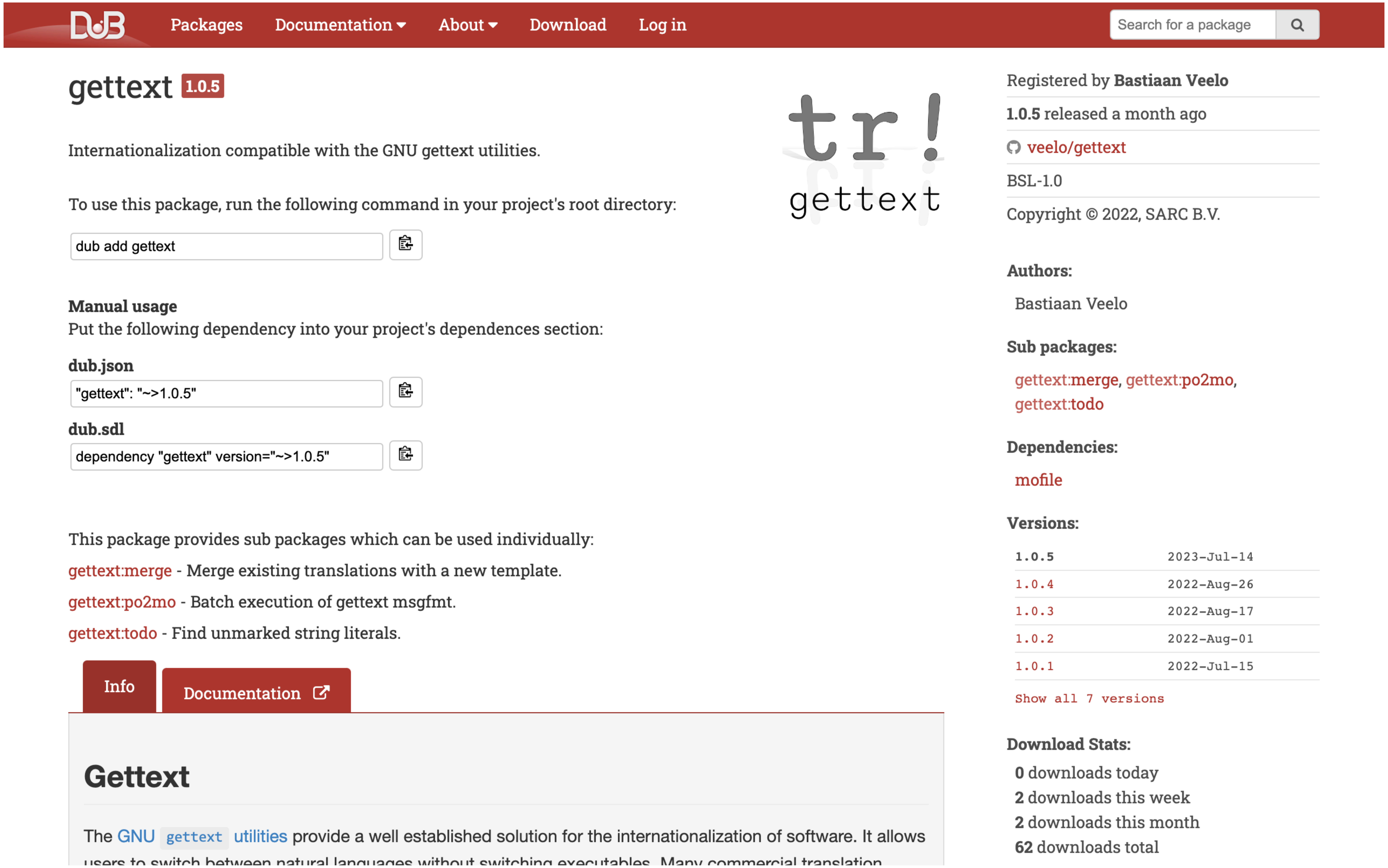
https://www.gnu.org/software/gettext/manual/

```
#include <libintl.h>

puts(gettext(“This is a translatable string.\n”));
printf(gettext("String \"%s\" has %d characters.\n”),
       s, strlen(s));
// TRANSLATORS: Comment regarding following string.
printf(pgettext("Message recipient", "Name: %s”),
       recipient_name);
printf(ngettext("One file copied", "%d files copied”, n), n);
```

https://
www.gnu.org/
software/
gettext/manual/

```c
#include <libintl.h>

puts(gettext(“This is a translatable string.\n”));
printf(gettext("String \"%s\" has %d characters.\n”),
       s, strlen(s));
// TRANSLATORS: Comment regarding following string.
printf(pgettext("Message recipient", "Name: %s”),
       recipient_name);
printf(ngettext("One file copied", "%d files copied”, n), n);

#define gettext_noop(String) String

static const char *messages[] = {
    gettext_noop("some very meaningful message"),
    gettext_noop("and another one")
};
const char *str;
str = index > 1 ?
    gettext("a default message") :
    gettext(messages[index]);
```

https://www.gnu.org/software/gettext/manual/



gettext 1.0.5

Internationalization compatible with the GNU gettext utilities.

To use this package, run the following command in your project's root directory:

```
dub add gettext
```

Manual usage
Put the following dependency into your project's dependences section:

dub.json

```
"gettext": "~>1.0.5"
```

dub.sdl

```
dependency "gettext" version="~>1.0.5"
```

This package provides sub packages which can be used individually:

gettext:merge - Merge existing translations with a new template.

gettext:po2mo - Batch execution of gettext msgfmt.

gettext:todo - Find unmarked string literals.

Gettext

The GNU `gettext` utilities provide a well established solution for the internationalization of software. It allows users to switch between natural languages without switching executables. Many commercial translation

Registered by **Bastiaan Veelo**

1.0.5 released a month ago

veelo/gettext

BSL-1.0

Copyright © 2022, SARC B.V.

Authors:

Bastiaan Veelo

Sub packages:

gettext:merge, gettext:po2mo, gettext:todo

Dependencies:

mofile

Versions:

1.0.5	2023-Jul-14
1.0.4	2022-Aug-26
1.0.3	2022-Aug-17
1.0.2	2022-Aug-01
1.0.1	2022-Jul-15

Show all 7 versions

Download Stats:

0 downloads today

2 downloads this week

2 downloads this month

62 downloads total

https://code.dlang.org/packages/gettext

```d
import gettext;

writeln(tr!"This is a translatable string.");
```

https://code.dlang.org/packages/gettext

```d
import gettext : _ = tr;

writeln(_!”This is a translatable string.");
```

https://
code.dlang.org/
packages/
gettext

https://code.dlang.org/packages/gettext

```
writeln(tr!"\tIndented by escape code.");
```

https://
code.dlang.org/
packages/
gettext

```d
writeln(tr!"\tIndented by escape code.");

writeln(tr!`Popular "wysiwyg" string.`);
```

https://
code.dlang.org/
packages/
gettext

```
writeln(tr!"\tIndented by escape code.");

writeln(tr!`Popular "wysiwyg" string.`);

writeln(tr!r"Same thing, less popular.”);
```

https://code.dlang.org/packages/gettext

```
writeln(tr!"\tIndented by escape code.");

writeln(tr!`Popular "wysiwyg" string.`);

writeln(tr!r"Same thing, less popular.”);

writeln(tr!q"<A delimited string>”); // etc.
```

https://code.dlang.org/packages/gettext

```d
writeln(tr!"\tIndented by escape code.");

writeln(tr!`Popular "wysiwyg" string.`);

writeln(tr!r"Same thing, less popular.”);

writeln(tr!q"<A delimited string>”); // etc.

writeln(tr!q"EOS
This
is a multi-line
heredoc string
EOS"
);
```

https://code.dlang.org/packages/gettext

```d
writeln(tr!"\tIndented by escape code.");

writeln(tr!`Popular "wysiwyg" string.`);

writeln(tr!r"Same thing, less popular.”);

writeln(tr!q"<A delimited string>”); // etc.

writeln(tr!q"EOS
This
is a multi-line
heredoc string
EOS"
);

writeln(tr!q{void foo();});
```

https://
code.dlang.org/
packages/
gettext

https://
code.dlang.org/
packages/
gettext

```
// Before:
writefln("%d green bottle(s) hanging on the wall", n);
```

https://
code.dlang.org/
packages/
gettext

```
// Before:
writefln("%d green bottle(s) hanging on the wall", n);
writefln(n == 1 ?
         "One green bottle hanging on the wall" :
         "%d green bottles hanging on the wall", n);
```

https://code.dlang.org/packages/gettext

```
// Before:
writefln("%d green bottle(s) hanging on the wall", n);
writefln(n == 1 ?
         "One green bottle hanging on the wall" :
         "%d green bottles hanging on the wall", n);


// After:
writeln(tr!("one green bottle hanging on the wall",
            "%d green bottles hanging on the wall")(n));
```

https://
code.dlang.org/
packages/
gettext

https://
code.dlang.org/
packages/
gettext

```
// Bad:
writeln(tr!"Welcome " ~ player ~ tr!", you may make a wish");
```

https://code.dlang.org/packages/gettext

```
// Bad:
writeln(tr!"Welcome " ~ player ~ tr!", you may make a wish");

// Good:
writefln(tr!("Welcome %s, you may make a wish"), player);
```

https://code.dlang.org/packages/gettext

```
// Bad:
writeln(tr!"Welcome " ~ player ~ tr!", you may make a wish");

// Good:
writefln(tr!("Welcome %s, you may make a wish"), player);

writefln(tr!("Welcome %s, you may make a wish",
             "Welcome %s, you may make %d wishes")(n), player);
```

https://
code.dlang.org/
packages/
gettext

https://code.dlang.org/packages/gettext

```d
foreach (i, where; ["hand", "bush"])
    writefln(i == 0 ?
             "%d bird in the %s" :
             "%d birds in the %s", i + 1, where);
```

https://
code.dlang.org/
packages/
gettext

```
foreach (i, where; ["hand", "bush"])
    writefln(i == 0 ?
             "%d bird in the %s" :
             "%d birds in the %s", i + 1, where);


foreach (i, where; [tr!"hand", tr!"bush"])
    writefln(tr!("One bird in the %1$s”,
                 "%2$d birds in the %1$s")(i + 1),
             where);
```

https://
code.dlang.org/
packages/
gettext

https://dlang.org/phobos/std_format.html

Format Strings

The functions contained in this package use *format strings*. A format string describes the layout of another string for reading or writing purposes. A format string is composed of normal text interspersed with *format specifiers*. A format specifier starts with a percentage sign '%', optionally followed by one or more *parameters* and ends with a *format indicator*. A format indicator may be a simple *format character* or a *compound indicator*.

Format strings are composed according to the following grammar:

```
FormatString:
    FormatStringItem FormatString
FormatStringItem:
    Character
    FormatSpecifier
FormatSpecifier:
    '%' Parameters FormatIndicator


FormatIndicator:
    FormatCharacter
    CompoundIndicator
FormatCharacter:
    see remark below
CompoundIndicator:
    '(' FormatString '%)'
    '(' FormatString '%|' Delimiter '%)'
Delimiter
    empty
    Character Delimiter


Parameters:
    Position Flags Width Precision Separator
```

https://dlang.org/
phobos/std_format.html

```
tr!("Walter Bright”, Comment("Proper name. Phonetically: ˈwɔltər braɪt”));
```

```
tr!("Walter Bright”, Comment("Proper name. Phonetically: ˈwɔltər braɪt”));


tr!("Review the draft.", Context("document"));
tr!("Review the draft.", Context("nautical"),
                         Comment(`Nautical term! "Draft" = how deep the bottom` ~
                                 `of the ship is below the water level.`));
```

```
tr!("Walter Bright”, Comment("Proper name. Phonetically: ˈwɔltər braɪt”));


tr!("Review the draft.", Context("document"));
tr!("Review the draft.", Context("nautical"),
                         Comment(`Nautical term! "Draft" = how deep the bottom` ~
                                 `of the ship is below the water level.`));


tr!("One license.", "%d licenses.", Context("software"),
                                    Comment("Notice to translator."))(n);
tr!("One license.", "%d licenses.", Context(“driver's"))(n);
```

```
static const magic = tr!"Compile time translation?!";
static const magic = tr!"Compile time translation?!";

enum {
    monday    = tr!"Monday",
    tuesday   = tr!"Tuesday",
    wednesday = tr!"Wednesday",
    thursday  = tr!"Thursday",
    friday    = tr!"Friday",
    saturday  = tr!"Saturday",
    sunday    = tr!"Sunday",
}
```

```
static const magic = tr!"Compile time translation?!";

enum {
    monday    = tr!"Monday",
    tuesday   = tr!"Tuesday",
    wednesday = tr!"Wednesday",
    thursday  = tr!"Thursday",
    friday    = tr!"Friday",
    saturday  = tr!"Saturday",
    sunday    = tr!"Sunday",
}

struct S
{
    auto day = monday;
    auto city = tr!"Gothenburg";
}
```

C

```c
#include <libintl.h>

#define gettext_noop(String) String

static const char *messages[] = {
    gettext_noop("some very meaningful message"),
    gettext_noop("and another one")
};
const char *str = index > 1 ?
                  gettext("a default message") :
                  gettext(messages[index]);
```

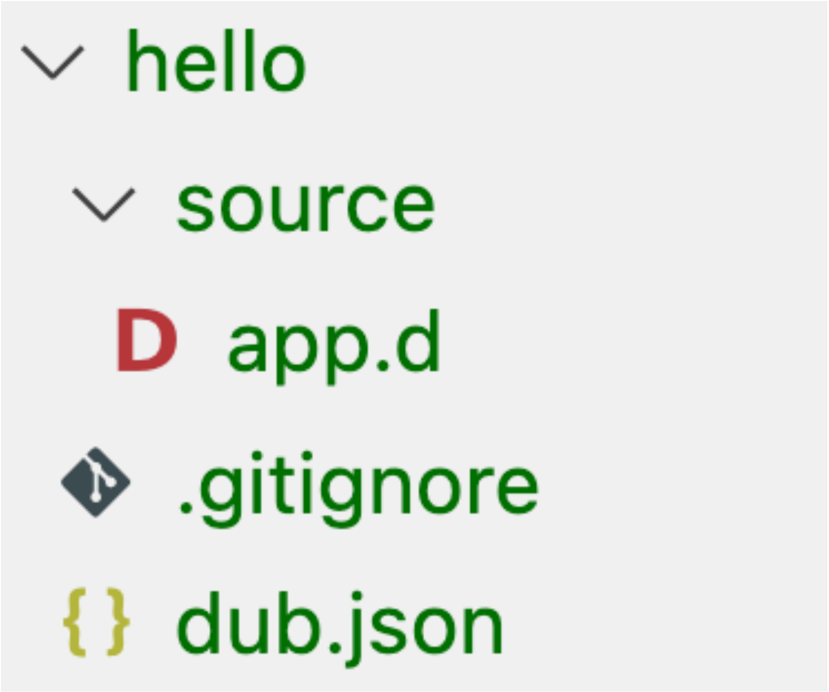
C

```
#include <libintl.h>

#define gettext_noop(String) String

static const char *messages[] = {
    gettext_noop("some very meaningful message"),
    gettext_noop("and another one")
};
const char *str = index > 1 ?
                  gettext("a default message") :
                  gettext(messages[index]);
```

D

```
import gettext;

static const messages = [tr!"some very meaningful message",
                         tr!"and another one"];
string str = index > 1 ?
             tr!"default message" :
             messages[index];
```

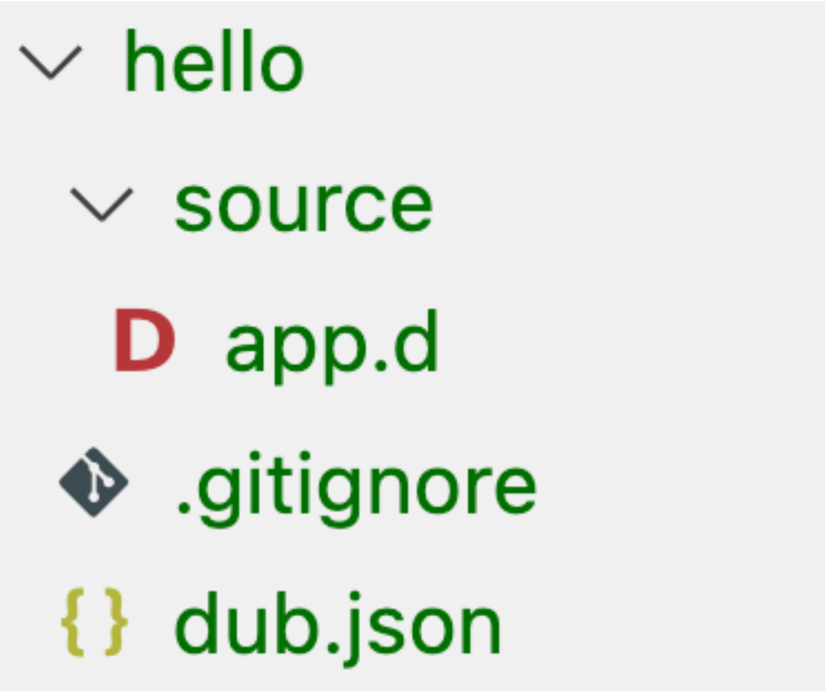
```
mixin(`writeln(tr!"This is mixed in code.");`);
```



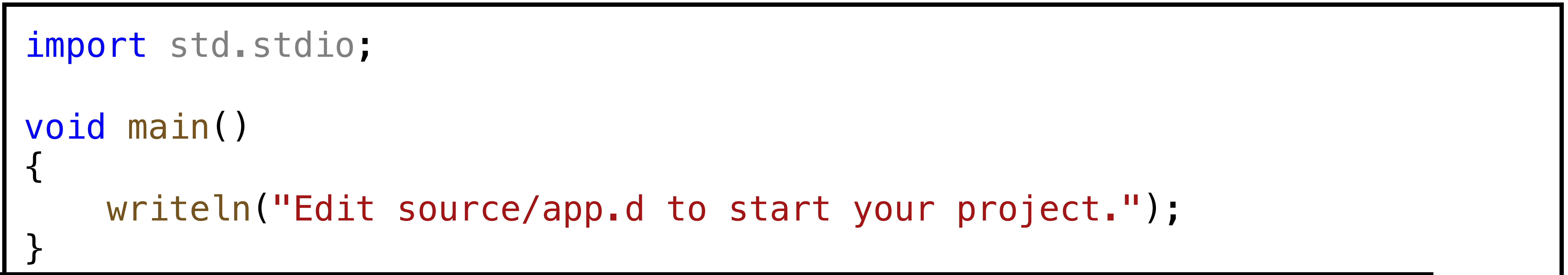
```
hello
  source
    app.d
  .gitignore
  dub.json
```

```d
import std.stdio;

void main()
{
    writeln("Edit source/app.d to start your project.");
}
```

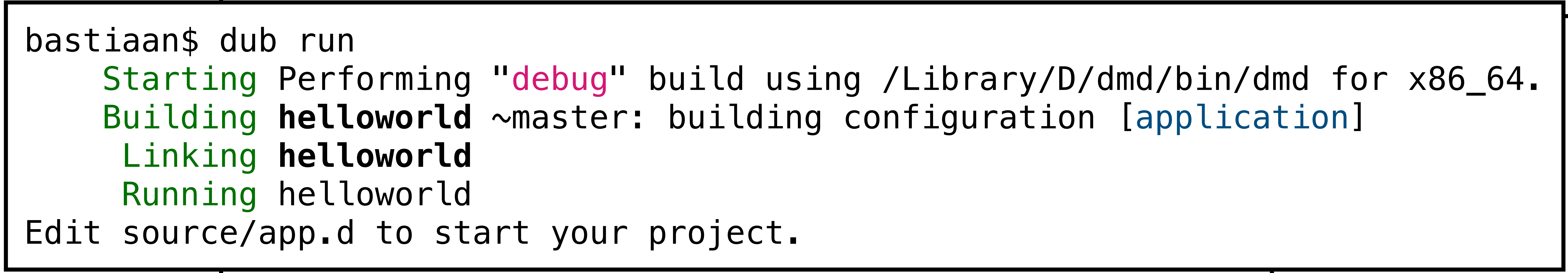
```json
{
    "authors": [
        "Bastiaan Veelo"
    ],
    "copyright": "Copyright © 2023, Bastiaan Veelo",
    "description": "A minimal D application.",
    "license": "proprietary",
    "name": "helloworld"
}
```



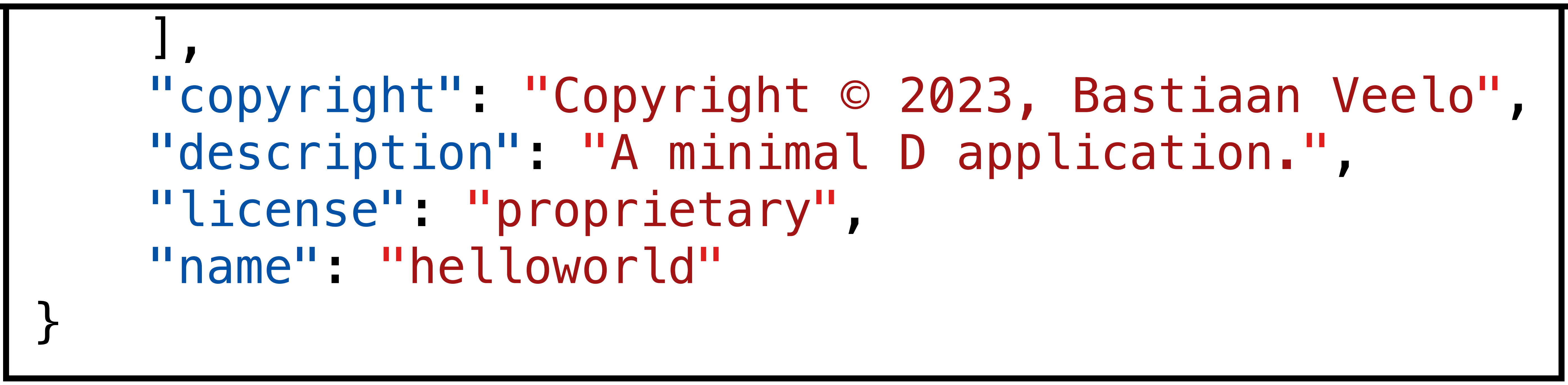
```
hello
  source
    app.d
  .gitignore
  dub.json
```



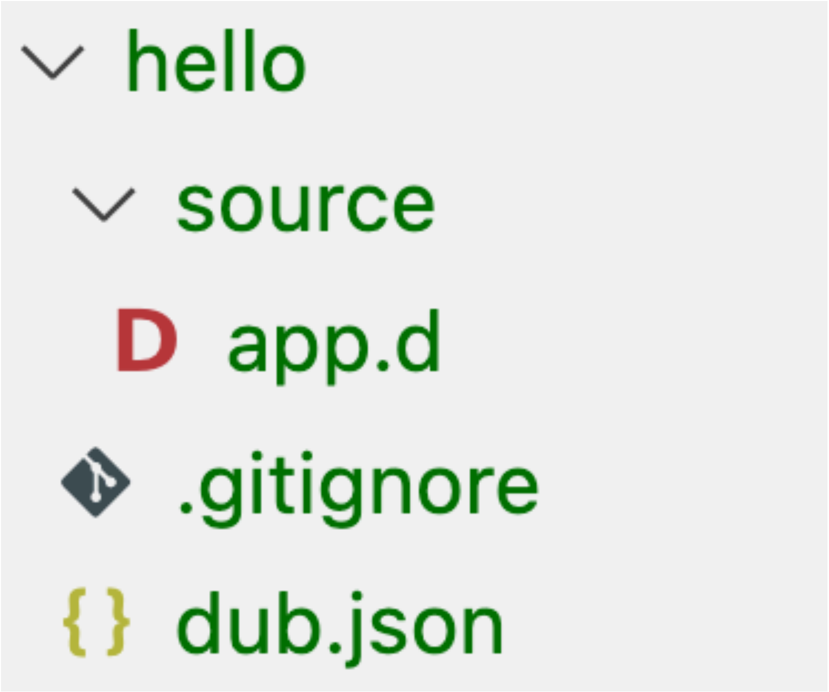
```d
import std.stdio;

void main()
{
    writeln("Edit source/app.d to start your project.");
}
```



```
bastiaan$ dub run
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
    Building helloworld ~master: building configuration [application]
     Linking helloworld
     Running helloworld
Edit source/app.d to start your project.
```



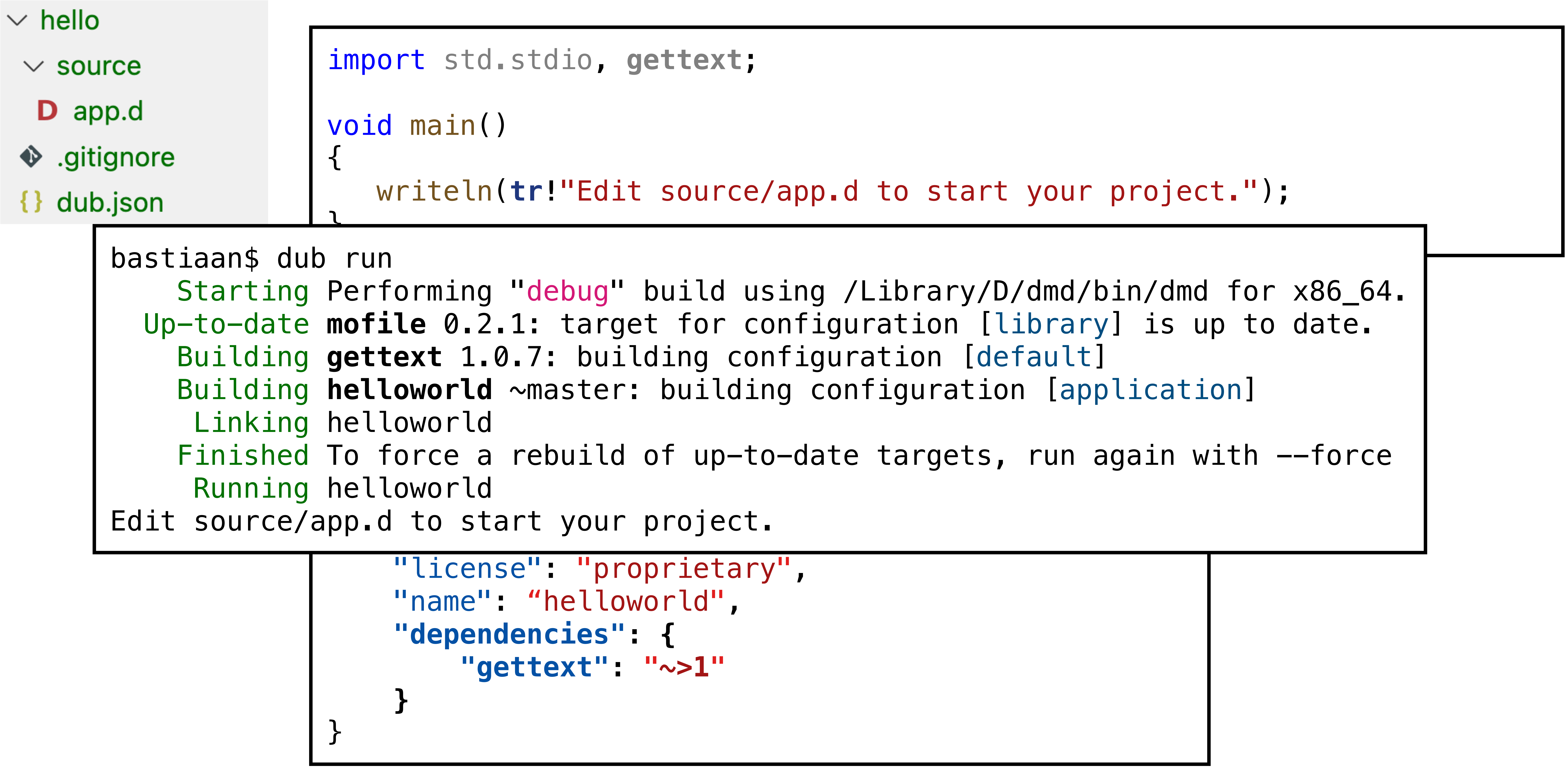
```json
    ],
    "copyright": "Copyright © 2023, Bastiaan Veelo",
    "description": "A minimal D application.",
    "license": "proprietary",
    "name": "helloworld"
}
```



```
hello
  source
    app.d
  .gitignore
  dub.json
```

```d
import std.stdio, gettext;

void main()
{
   writeln(tr!"Edit source/app.d to start your project.");
}
```

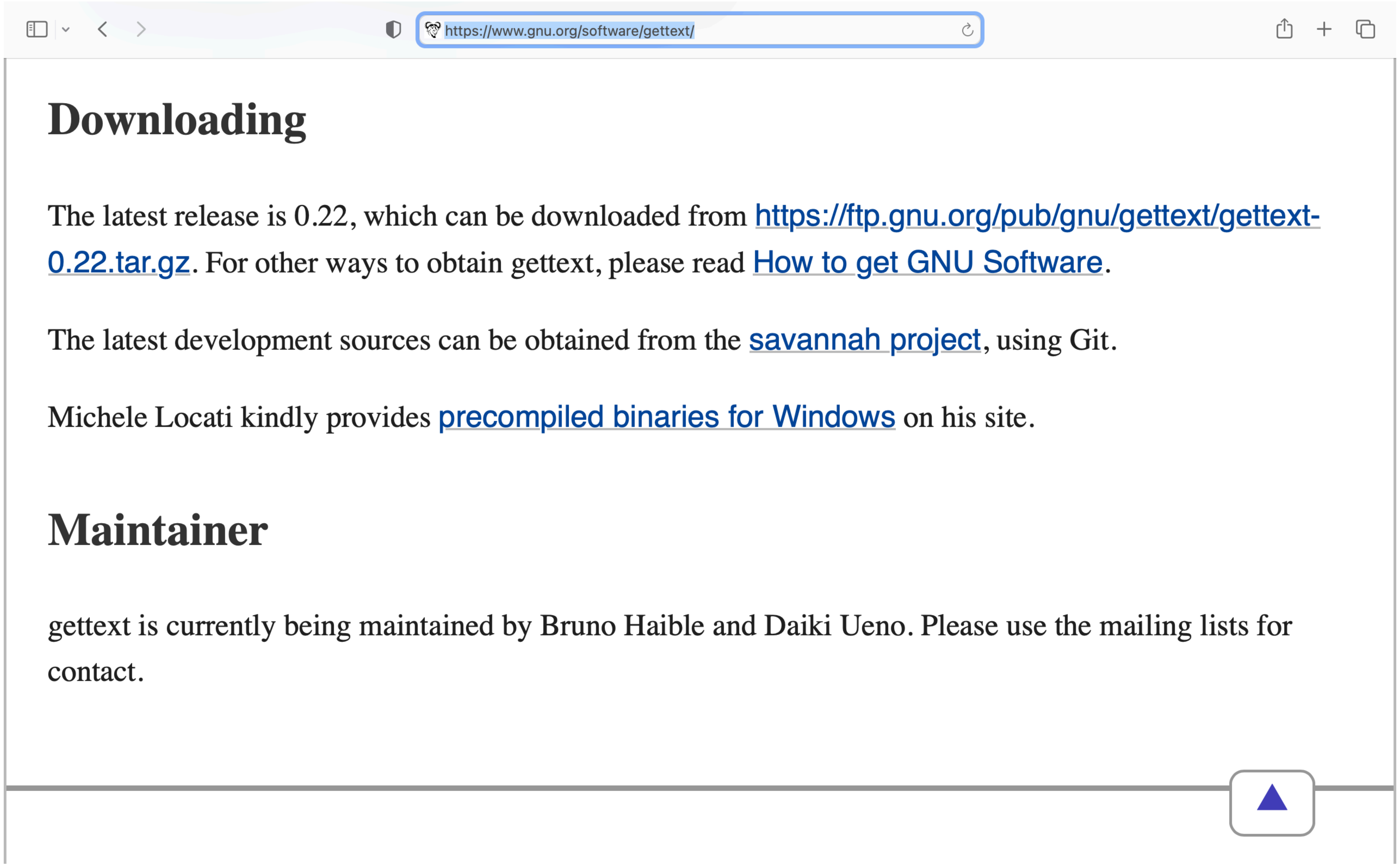
```json
{
    "authors": [
        "Bastiaan Veelo"
    ],
    "copyright": "Copyright © 2023, Bastiaan Veelo",
    "description": "A minimal D application.",
    "license": "proprietary",
    "name": “helloworld",
    "dependencies": {
        "gettext": "~>1"
    }
}
```



```d
import std.stdio, gettext;

void main()
{
   writeln(tr!"Edit source/app.d to start your project.");
}
```

```
bastiaan$ dub run
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
    Building gettext 1.0.7: building configuration [default]
    Building helloworld ~master: building configuration [application]
     Linking helloworld
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running helloworld
Edit source/app.d to start your project.
```

```json
    "license": "proprietary",
    "name": “helloworld",
    "dependencies": {
        "gettext": "~>1"
    }
}
```



Downloading

The latest release is 0.22, which can be downloaded from https://ftp.gnu.org/pub/gnu/gettext/gettext-0.22.tar.gz. For other ways to obtain gettext, please read How to get GNU Software.

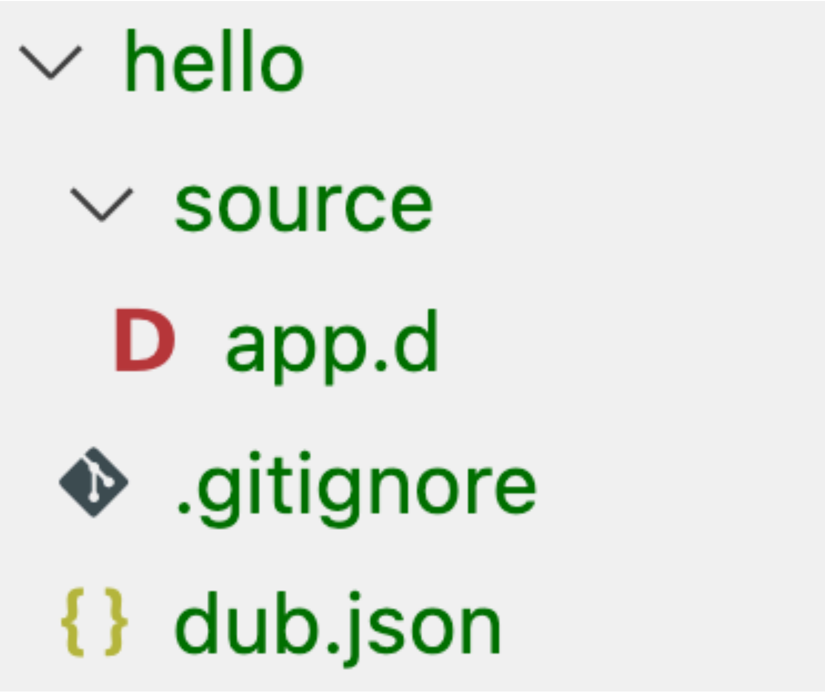
The latest development sources can be obtained from the savannah project, using Git.

Michele Locati kindly provides precompiled binaries for Windows on his site.

Maintainer

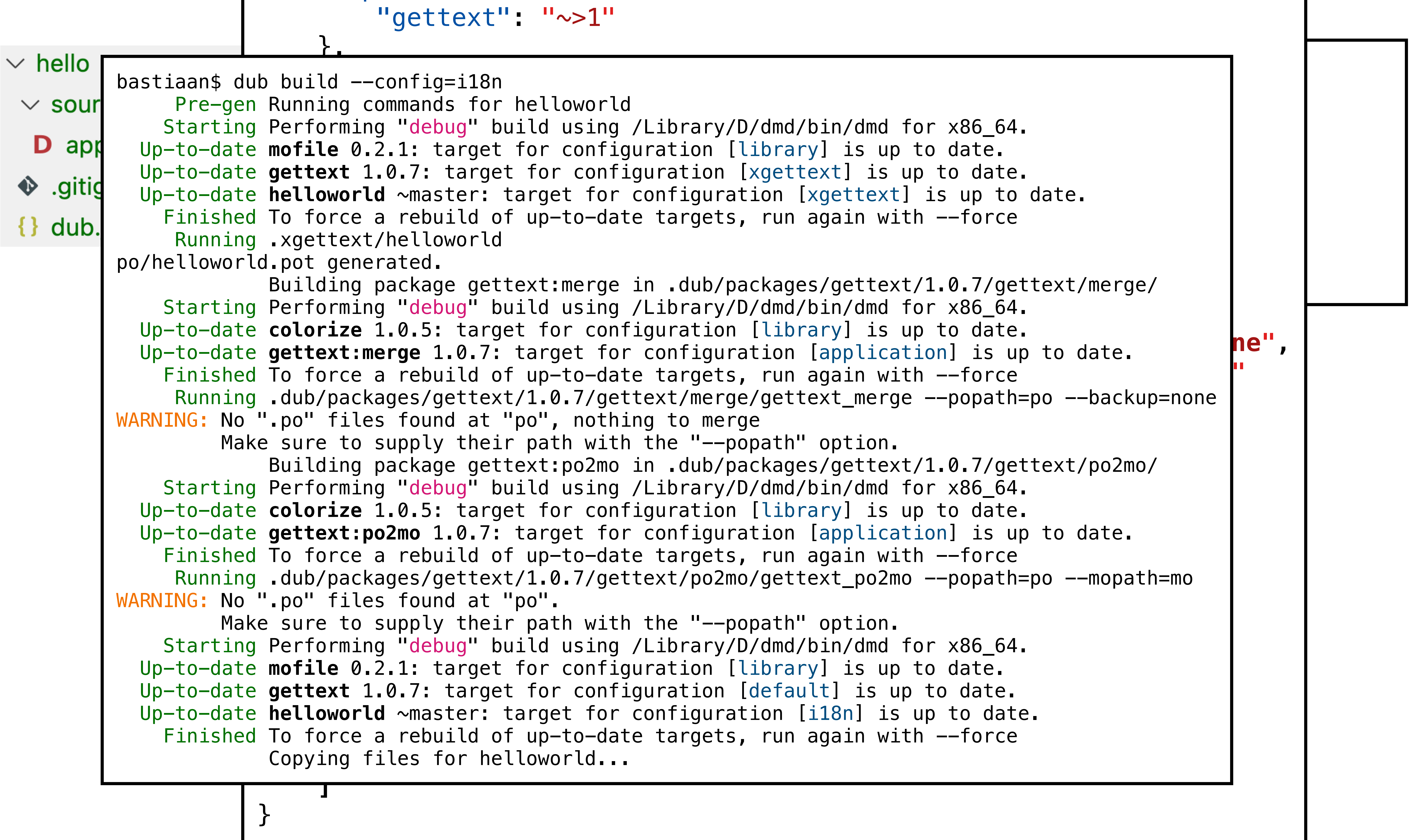
gettext is currently being maintained by Bruno Haible and Daiki Ueno. Please use the mailing lists for contact.

https://
www.gnu.org/
software/
gettext/



- hello
  - source
    - app.d
  - .gitignore
  - dub.json

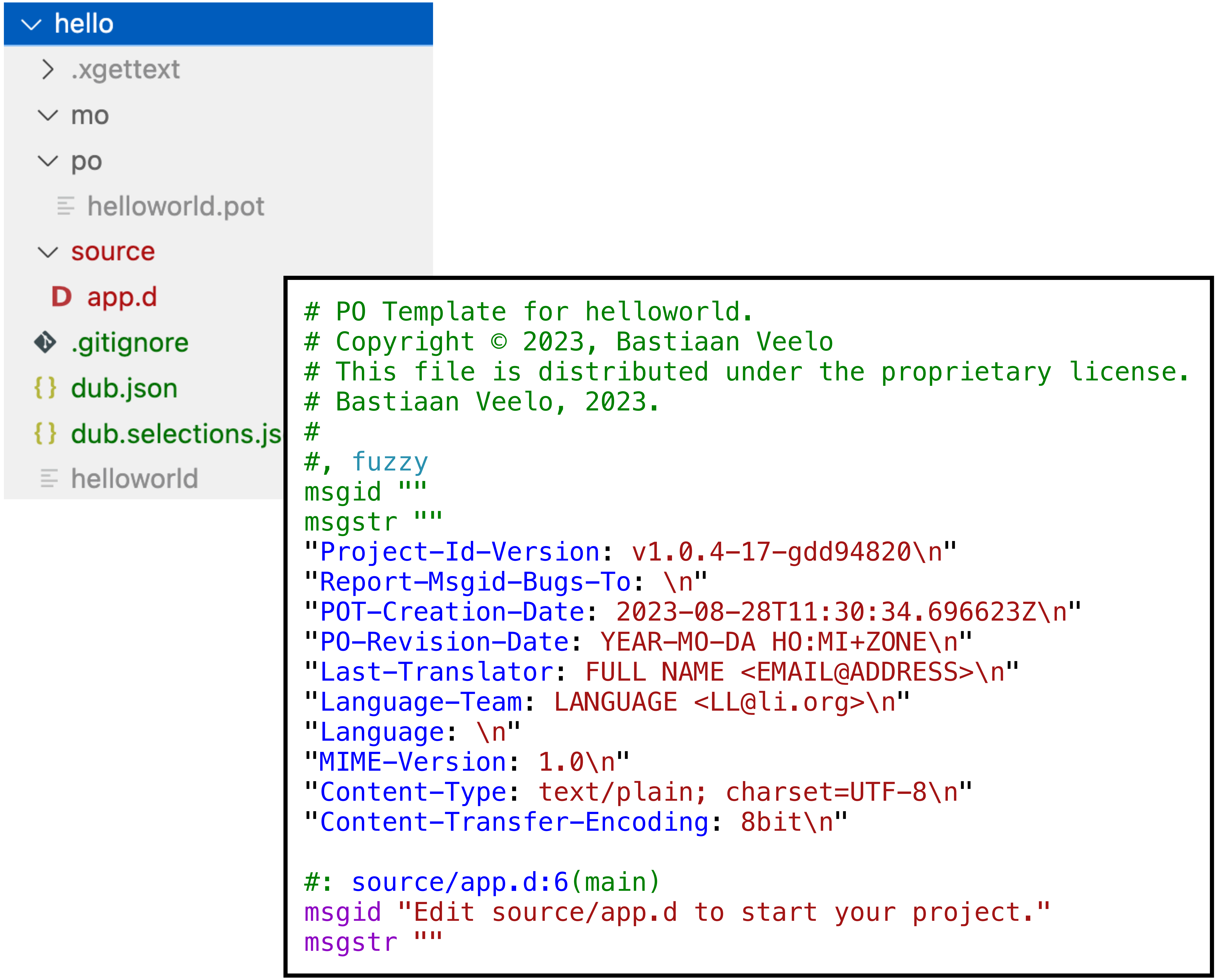
```
        "gettext": "~>1"
    },
    "targetType": "executable",
    "configurations": [
        {
            "name": "default"
        },
        {
            "name": "i18n",
            "preGenerateCommands": [
                "dub run --config=xgettext",
                "dub run gettext:merge -- --popath=po --backup=none",
                "dub run gettext:po2mo -- --popath=po --mopath=mo"
            ],
            "copyFiles": [
                "mo"
            ]
        },
        {
            "name": "xgettext",
            "targetPath": ".xgettext",
            "versions": [ "xgettext" ],
            "subConfigurations": {
                "gettext": "xgettext"
            }
        }
    ]
}
```



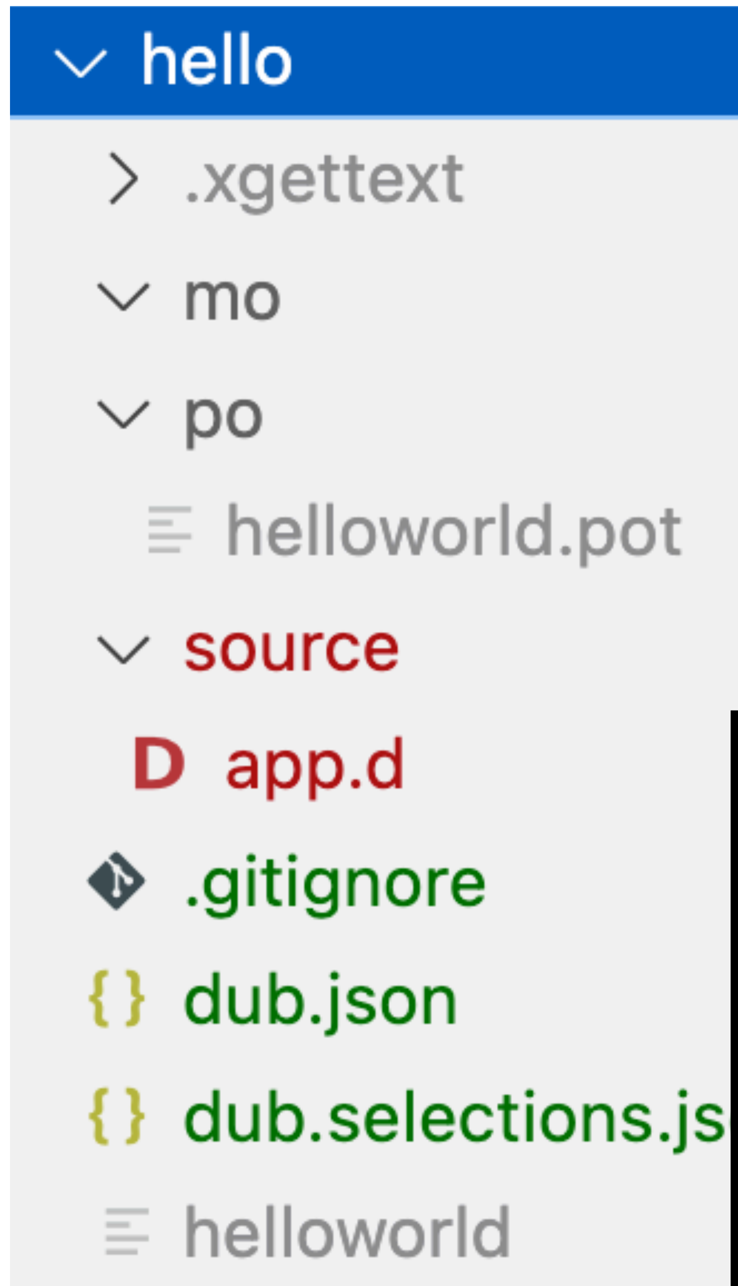
```
    "gettext": "~>1"
},
```

```
bastiaan$ dub build --config=i18n
     Pre-gen Running commands for helloworld
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
  Up-to-date gettext 1.0.7: target for configuration [xgettext] is up to date.
  Up-to-date helloworld ~master: target for configuration [xgettext] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running .xgettext/helloworld
po/helloworld.pot generated.
             Building package gettext:merge in .dub/packages/gettext/1.0.7/gettext/merge/
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date colorize 1.0.5: target for configuration [library] is up to date.
  Up-to-date gettext:merge 1.0.7: target for configuration [application] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running .dub/packages/gettext/1.0.7/gettext/merge/gettext_merge --popath=po --backup=none
WARNING: No ".po" files found at "po", nothing to merge
         Make sure to supply their path with the "--popath" option.
             Building package gettext:po2mo in .dub/packages/gettext/1.0.7/gettext/po2mo/
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date colorize 1.0.5: target for configuration [library] is up to date.
  Up-to-date gettext:po2mo 1.0.7: target for configuration [application] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running .dub/packages/gettext/1.0.7/gettext/po2mo/gettext_po2mo --popath=po --mopath=mo
WARNING: No ".po" files found at "po".
         Make sure to supply their path with the "--popath" option.
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
  Up-to-date gettext 1.0.7: target for configuration [default] is up to date.
  Up-to-date helloworld ~master: target for configuration [i18n] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
             Copying files for helloworld...
```

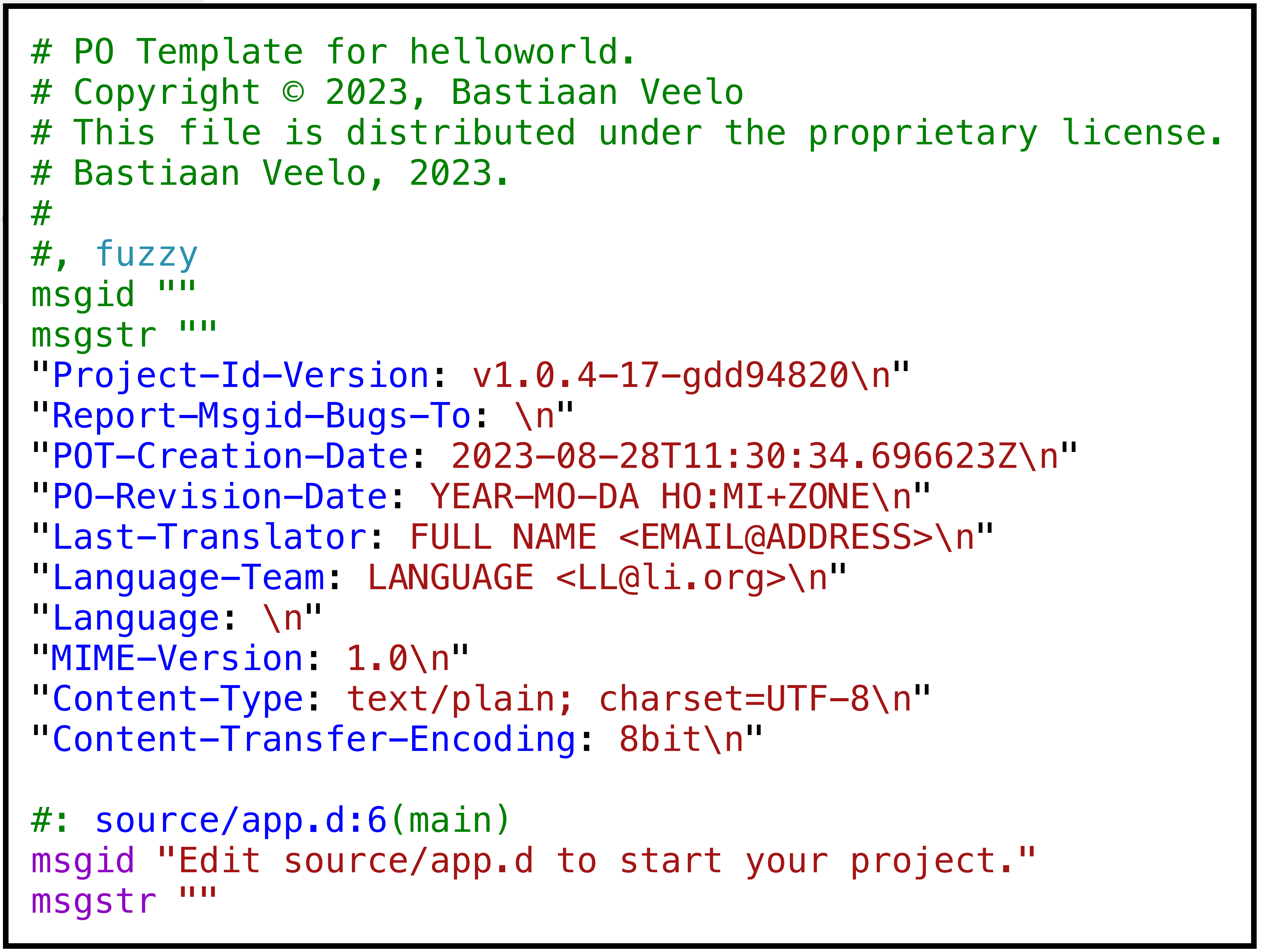
```
ne",
"
    ]
}
```



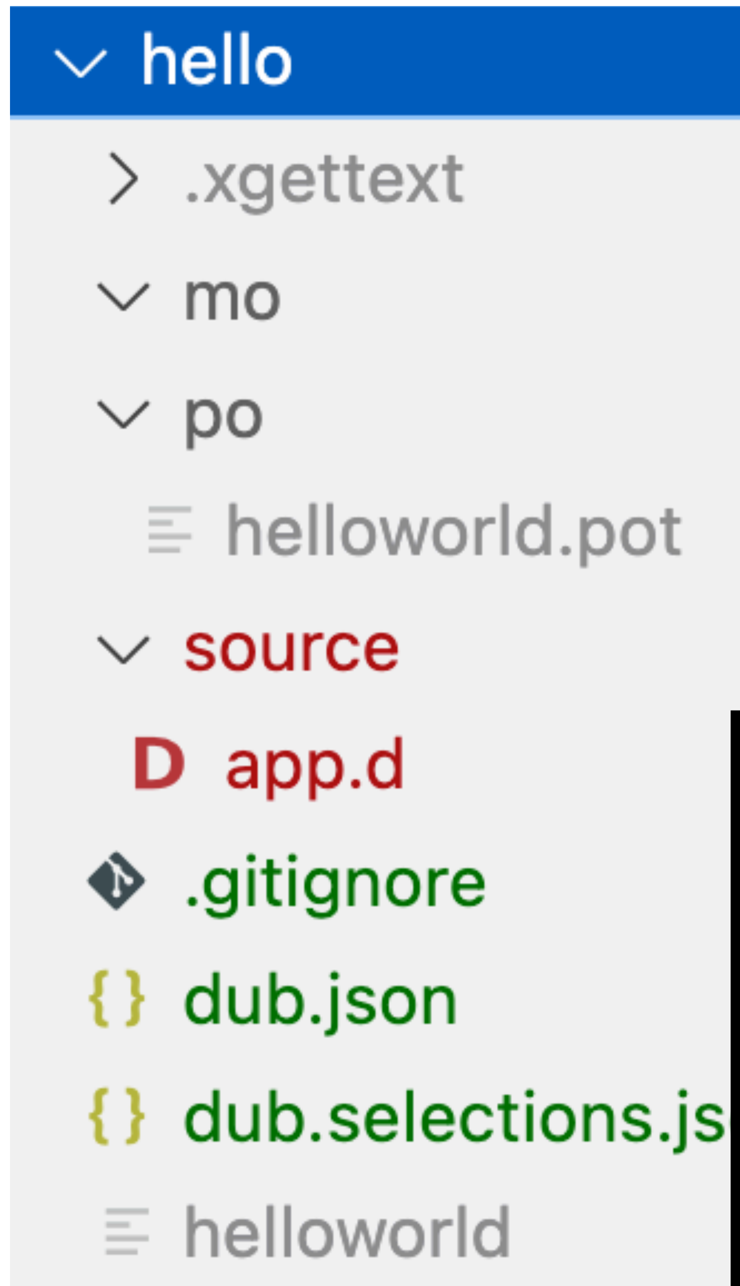
```
# PO Template for helloworld.
# Copyright © 2023, Bastiaan Veelo
# This file is distributed under the proprietary license.
# Bastiaan Veelo, 2023.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: v1.0.4-17-gdd94820\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2023-08-28T11:30:34.696623Z\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: source/app.d:6(main)
msgid "Edit source/app.d to start your project."
msgstr ""
```



```
msginit -i po/helloworld.pot -o po/nl_NL.po -l nl_NL
```

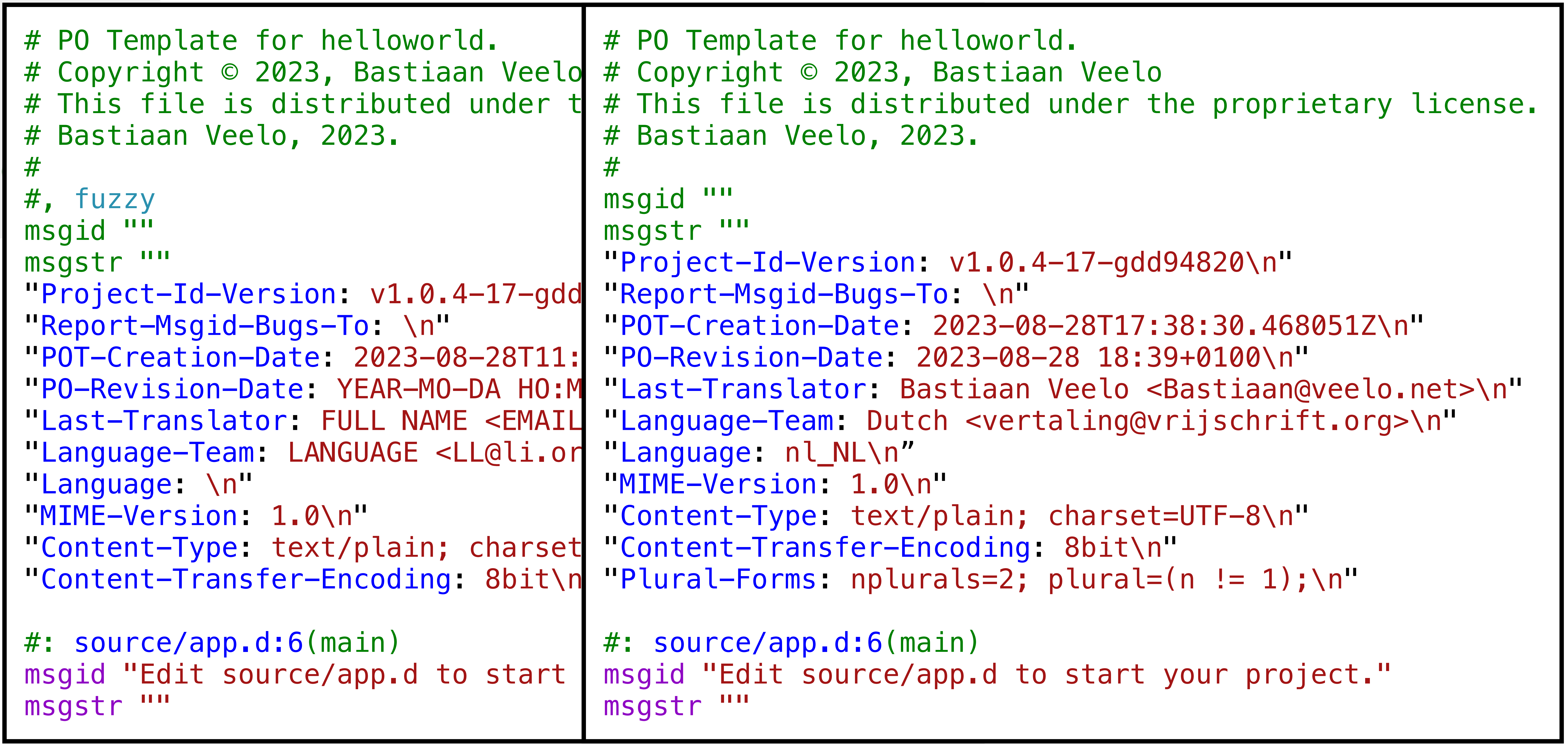


```
# PO Template for helloworld.
# Copyright © 2023, Bastiaan Veelo
# This file is distributed under the proprietary license.
# Bastiaan Veelo, 2023.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: v1.0.4-17-gdd94820\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2023-08-28T11:30:34.696623Z\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: source/app.d:6(main)
msgid "Edit source/app.d to start your project."
msgstr ""
```



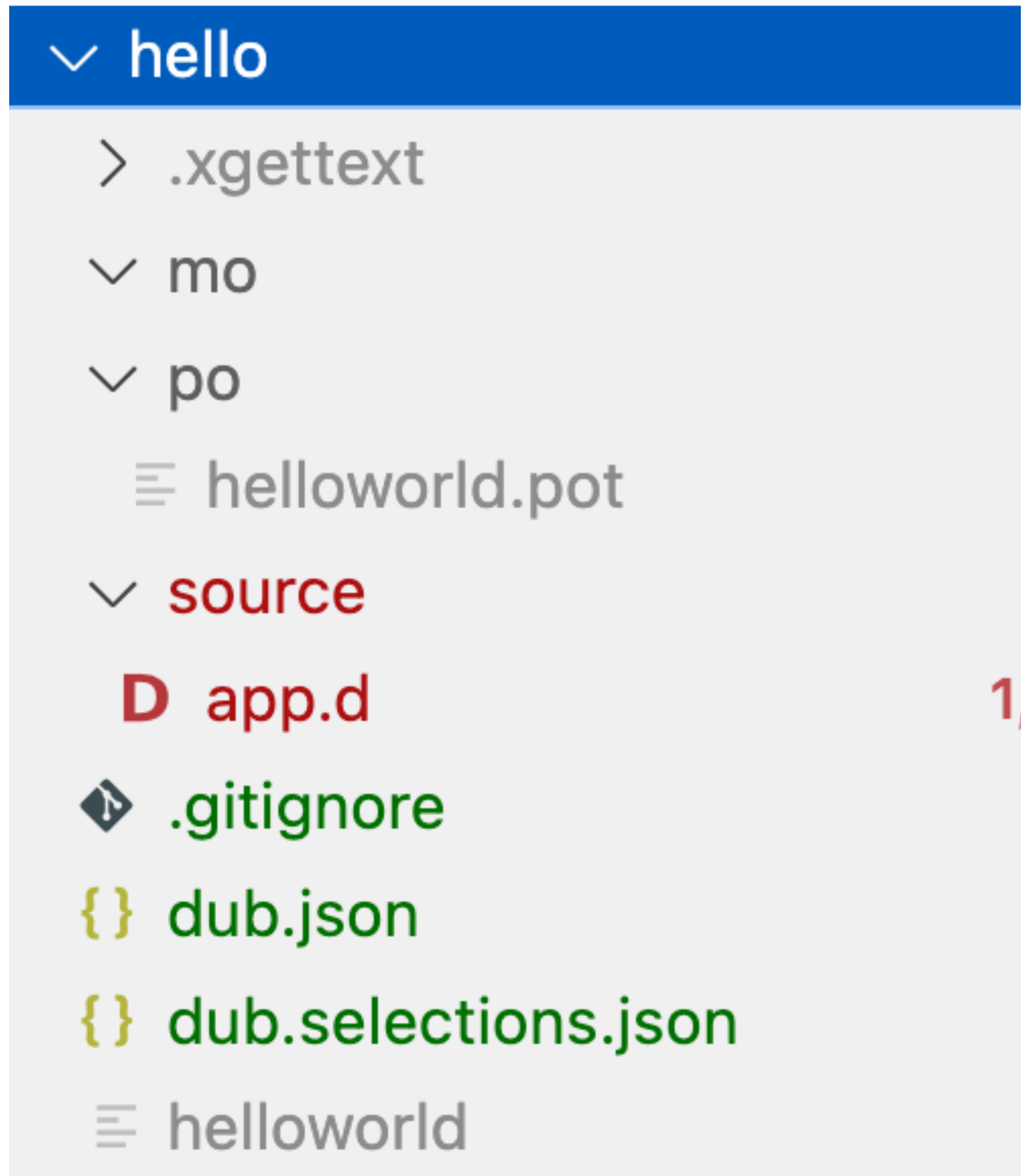
```
hello
  .xgettext
  mo
  po
    helloworld.pot
  source
    app.d
  .gitignore
  dub.json
  dub.selections.js
  helloworld
```

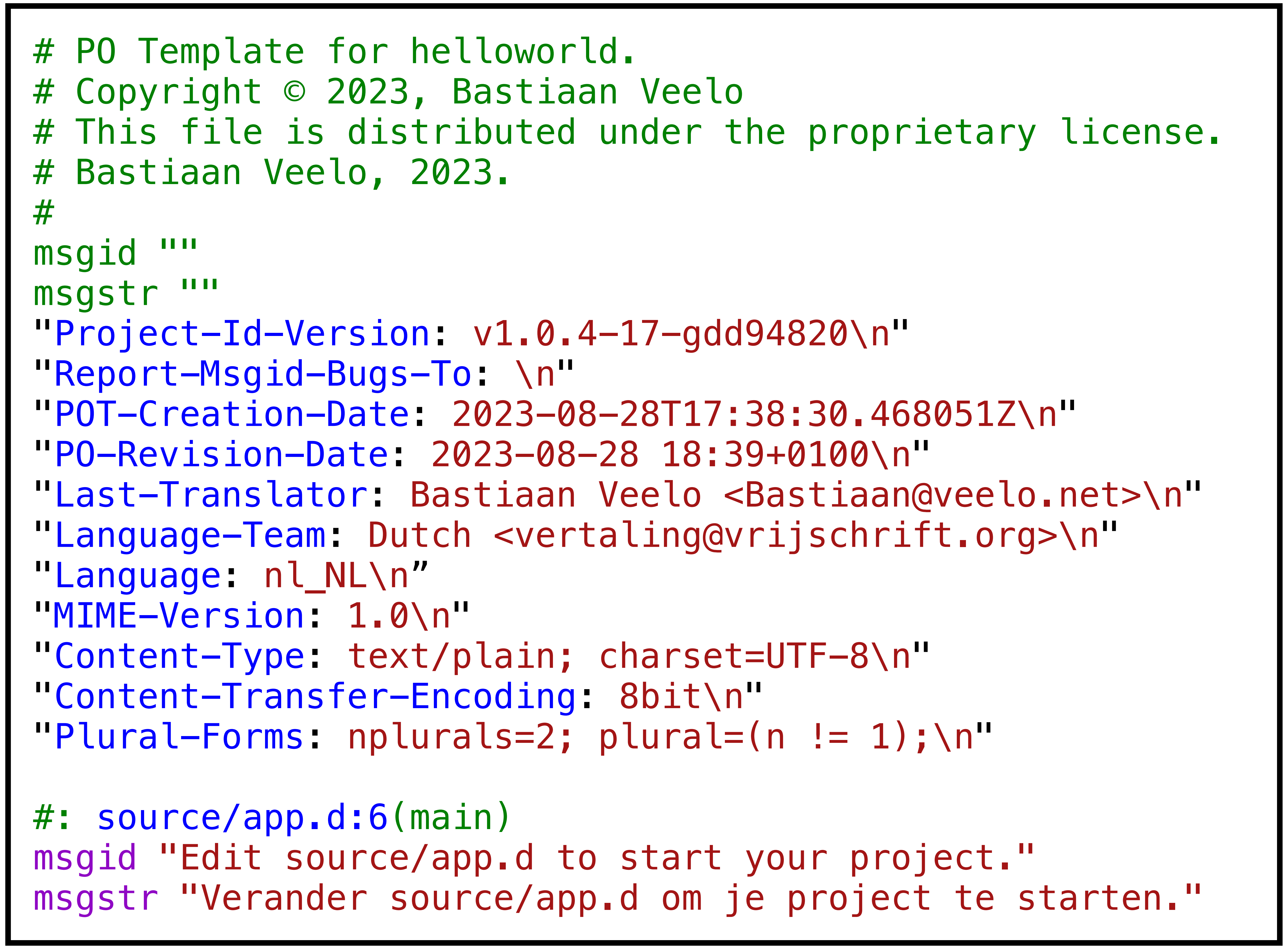
```
msginit -i po/helloworld.pot -o po/nl_NL.po -l nl_NL
```



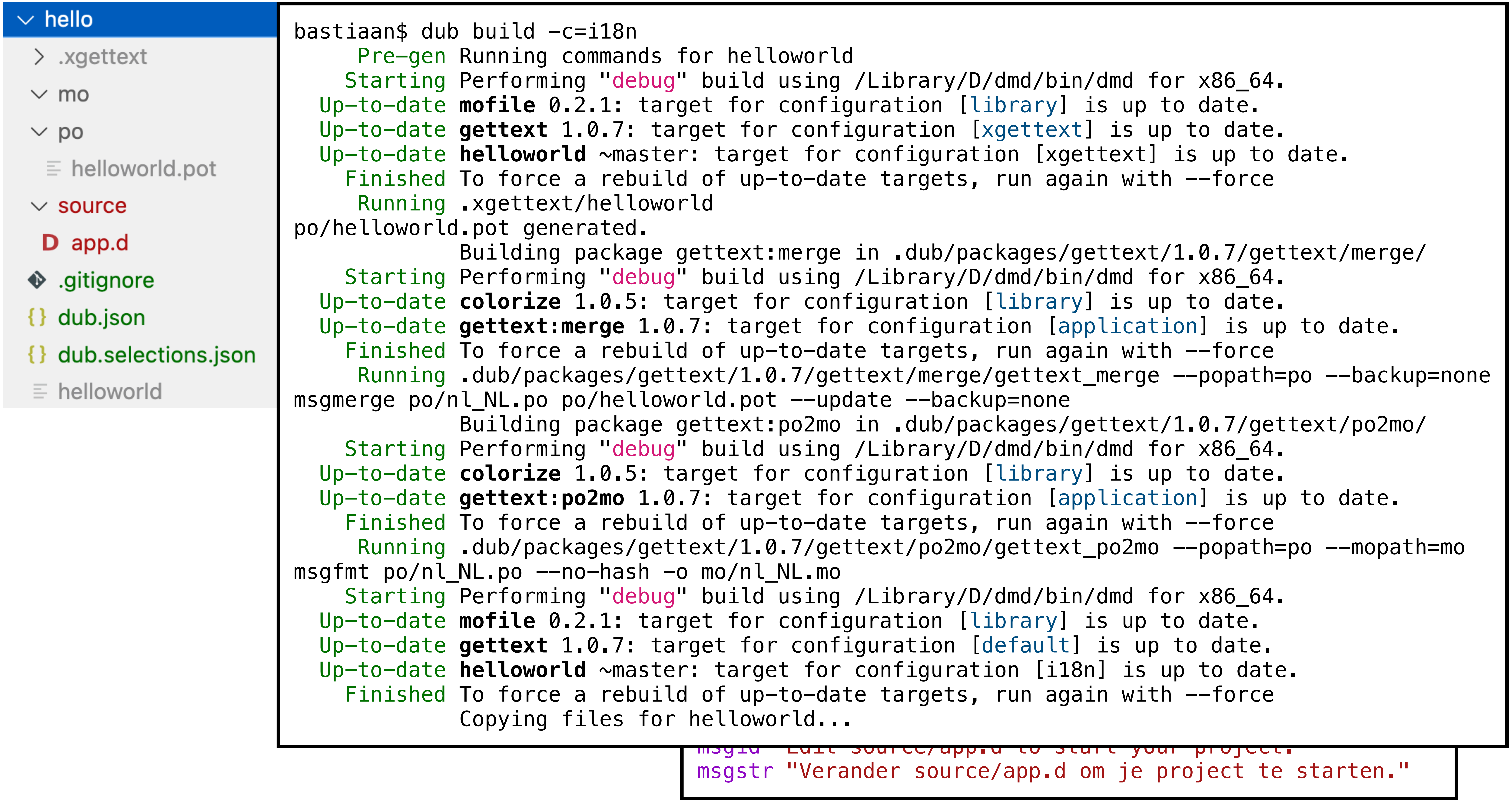
```
# PO Template for helloworld.
# Copyright © 2023, Bastiaan Veelo
# This file is distributed under t
# Bastiaan Veelo, 2023.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: v1.0.4-17-gdd
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2023-08-28T11:
"PO-Revision-Date: YEAR-MO-DA HO:M
"Last-Translator: FULL NAME <EMAIL
"Language-Team: LANGUAGE <LL@li.or
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset
"Content-Transfer-Encoding: 8bit\n

#: source/app.d:6(main)
msgid "Edit source/app.d to start
msgstr ""
```

```
# PO Template for helloworld.
# Copyright © 2023, Bastiaan Veelo
# This file is distributed under the proprietary license.
# Bastiaan Veelo, 2023.
#
msgid ""
msgstr ""
"Project-Id-Version: v1.0.4-17-gdd94820\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2023-08-28T17:38:30.468051Z\n"
"PO-Revision-Date: 2023-08-28 18:39+0100\n"
"Last-Translator: Bastiaan Veelo <Bastiaan@veelo.net>\n"
"Language-Team: Dutch <vertaling@vrijschrift.org>\n"
"Language: nl_NL\n”
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"

#: source/app.d:6(main)
msgid "Edit source/app.d to start your project."
msgstr ""
```



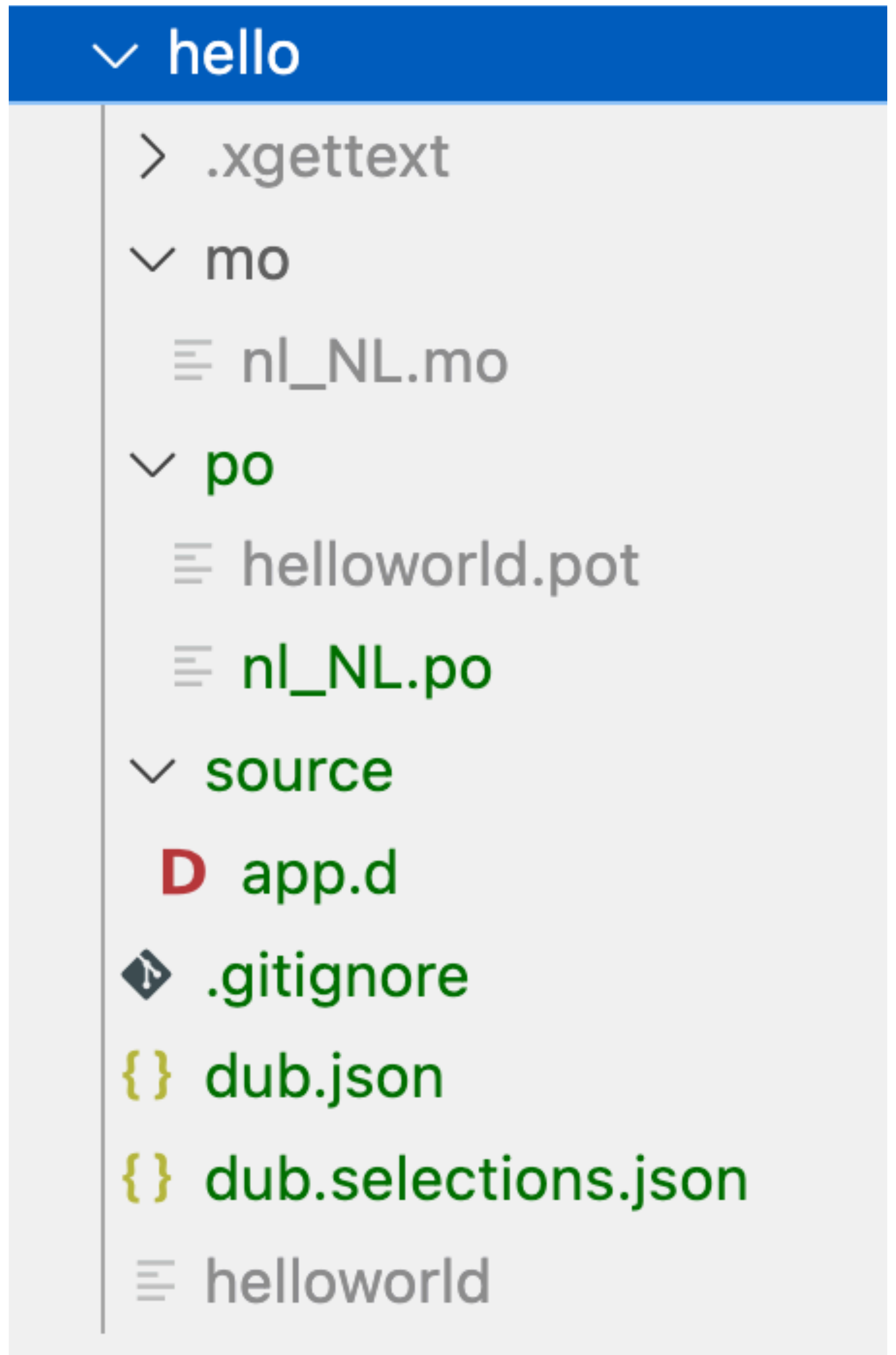


```
# PO Template for helloworld.
# Copyright © 2023, Bastiaan Veelo
# This file is distributed under the proprietary license.
# Bastiaan Veelo, 2023.
#
msgid ""
msgstr ""
"Project-Id-Version: v1.0.4-17-gdd94820\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2023-08-28T17:38:30.468051Z\n"
"PO-Revision-Date: 2023-08-28 18:39+0100\n"
"Last-Translator: Bastiaan Veelo <Bastiaan@veelo.net>\n"
"Language-Team: Dutch <vertaling@vrijschrift.org>\n"
"Language: nl_NL\n”
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"

#: source/app.d:6(main)
msgid "Edit source/app.d to start your project."
msgstr "Verander source/app.d om je project te starten."
```

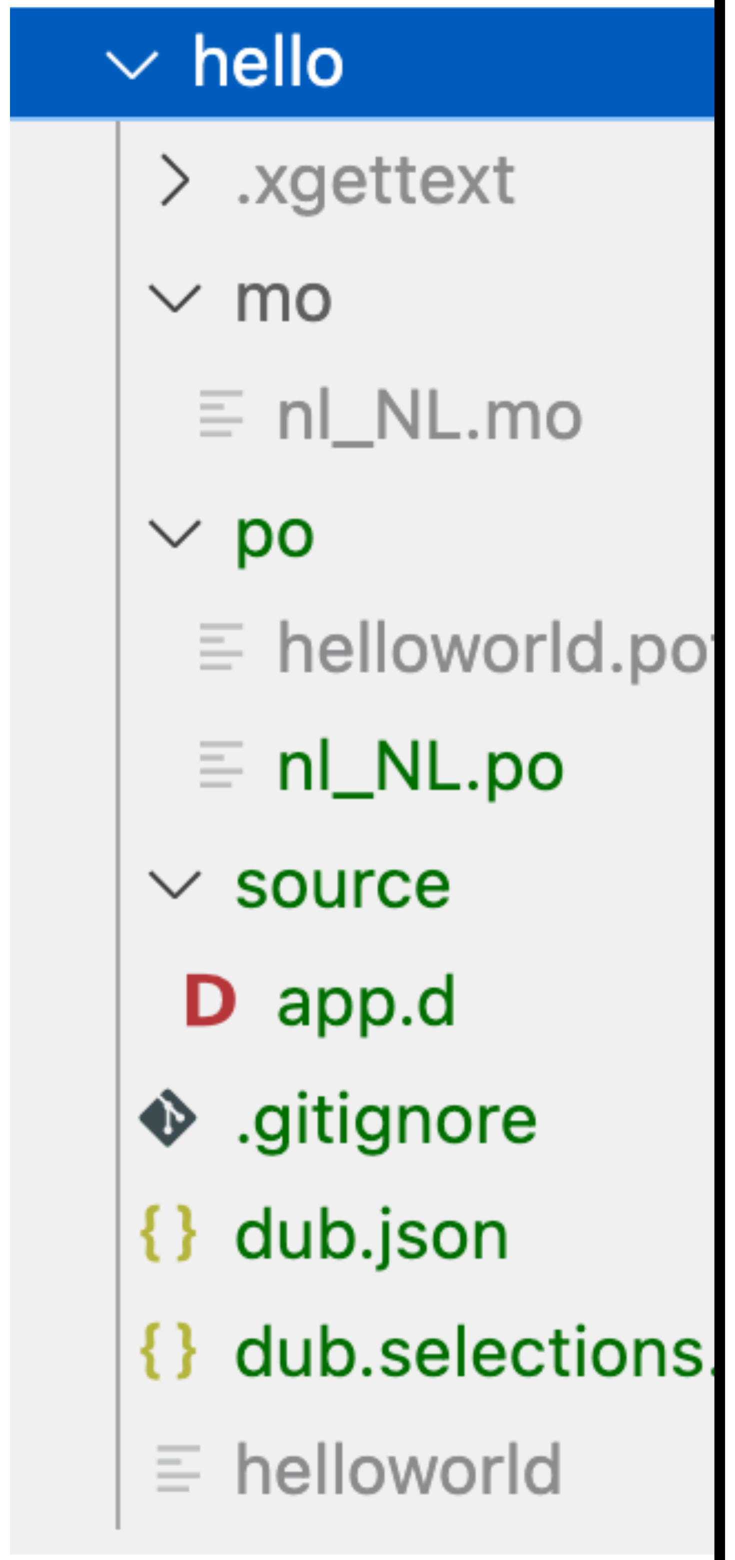


```
bastiaan$ dub build -c=i18n
     Pre-gen Running commands for helloworld
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
  Up-to-date gettext 1.0.7: target for configuration [xgettext] is up to date.
  Up-to-date helloworld ~master: target for configuration [xgettext] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running .xgettext/helloworld
po/helloworld.pot generated.
             Building package gettext:merge in .dub/packages/gettext/1.0.7/gettext/merge/
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date colorize 1.0.5: target for configuration [library] is up to date.
  Up-to-date gettext:merge 1.0.7: target for configuration [application] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running .dub/packages/gettext/1.0.7/gettext/merge/gettext_merge --popath=po --backup=none
msgmerge po/nl_NL.po po/helloworld.pot --update --backup=none
             Building package gettext:po2mo in .dub/packages/gettext/1.0.7/gettext/po2mo/
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date colorize 1.0.5: target for configuration [library] is up to date.
  Up-to-date gettext:po2mo 1.0.7: target for configuration [application] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running .dub/packages/gettext/1.0.7/gettext/po2mo/gettext_po2mo --popath=po --mopath=mo
msgfmt po/nl_NL.po --no-hash -o mo/nl_NL.mo
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
  Up-to-date gettext 1.0.7: target for configuration [default] is up to date.
  Up-to-date helloworld ~master: target for configuration [i18n] is up to date.
    Finished To force a rebuild of up-to-date targets, run again with --force
             Copying files for helloworld...
```

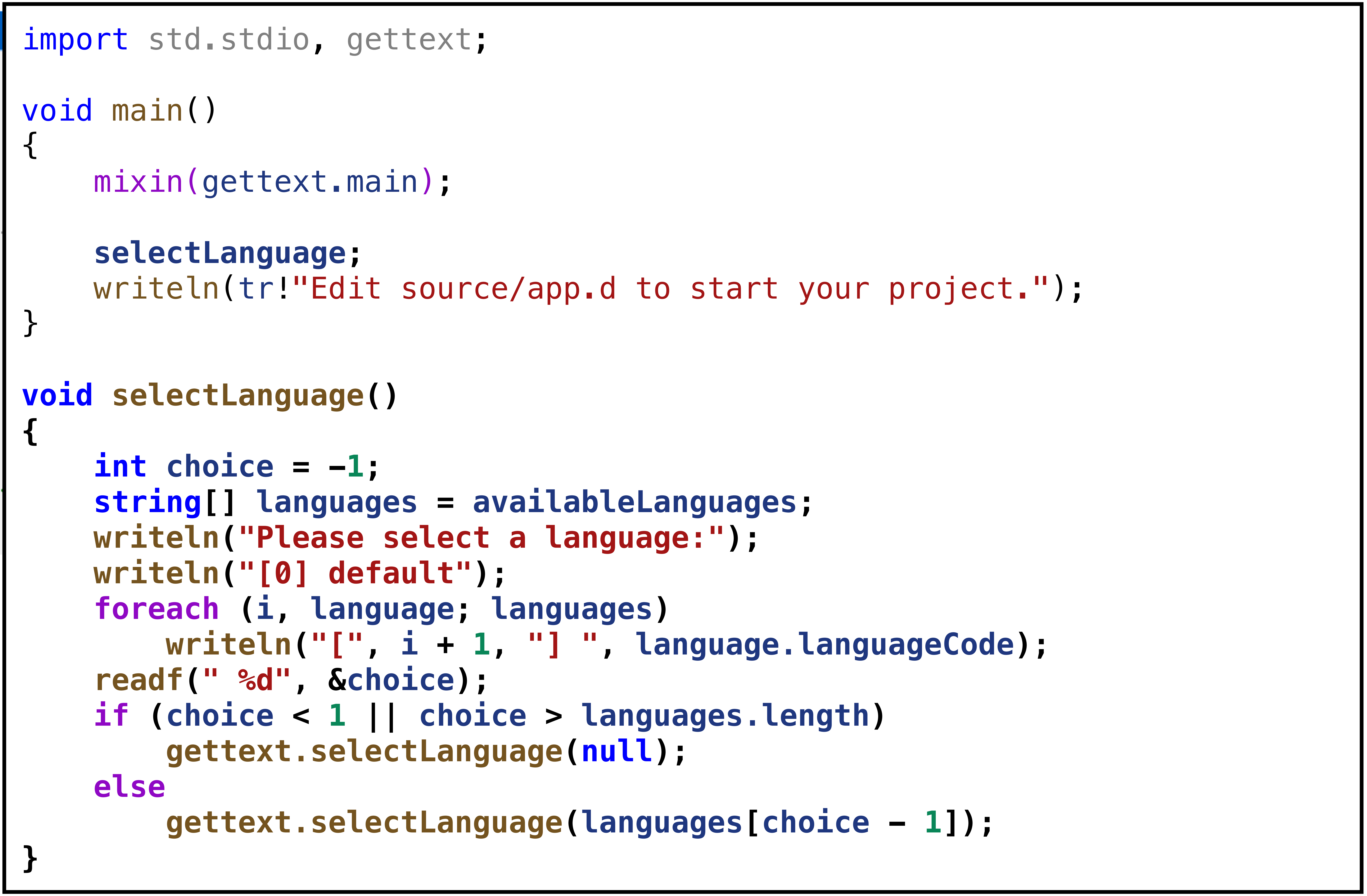
```
msgid "Edit source/app.d to start your project."
msgstr "Verander source/app.d om je project te starten."
```



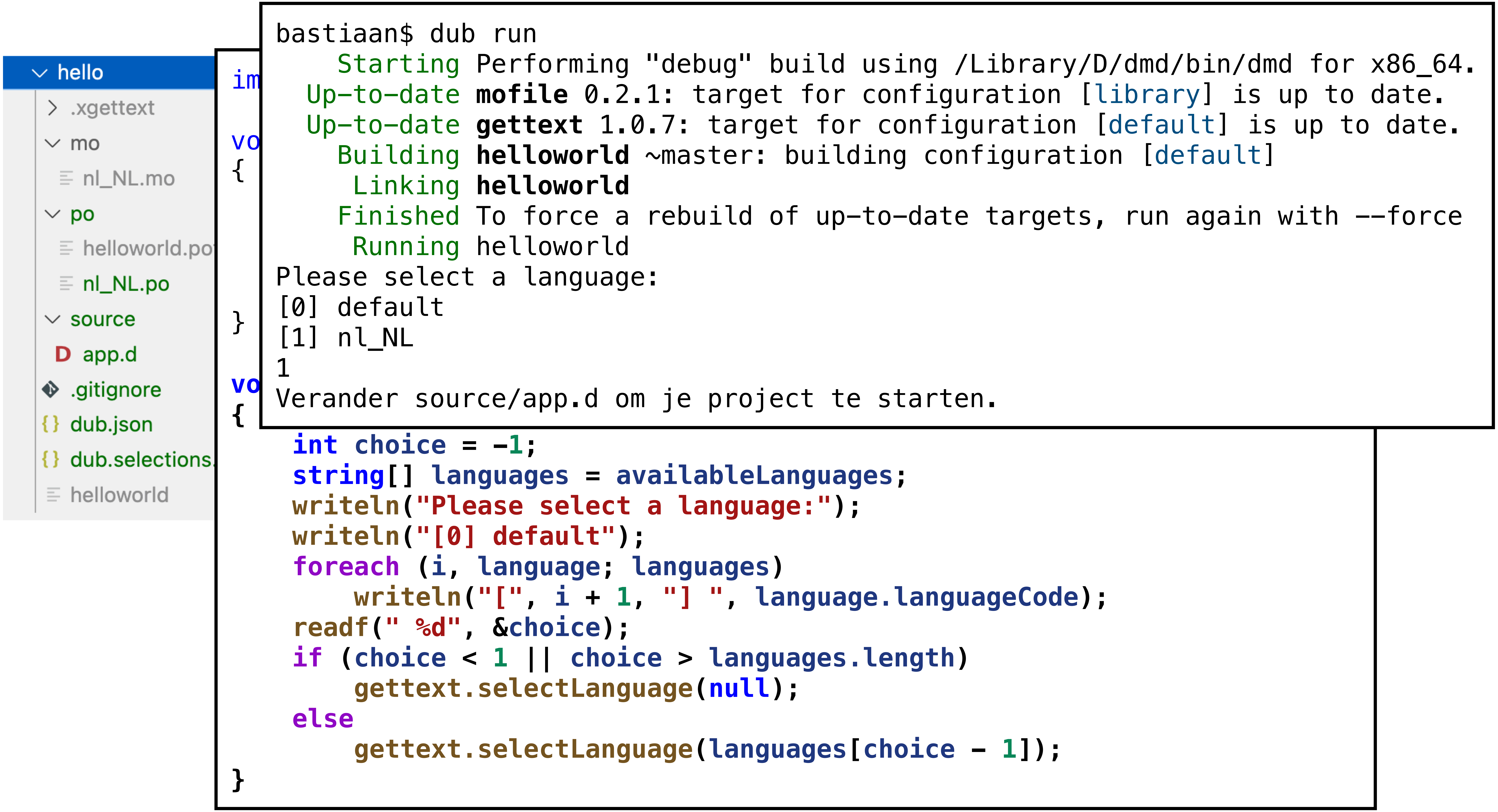
```
hello
  .xgettext
  mo
    nl_NL.mo
  po
    helloworld.pot
    nl_NL.po
  source
    app.d
  .gitignore
  dub.json
  dub.selections.json
  helloworld
```



```
hello
  .xgettext
  mo
    nl_NL.mo
  po
    helloworld.po
    nl_NL.po
  source
    app.d
  .gitignore
  dub.json
  dub.selections.
  helloworld
```

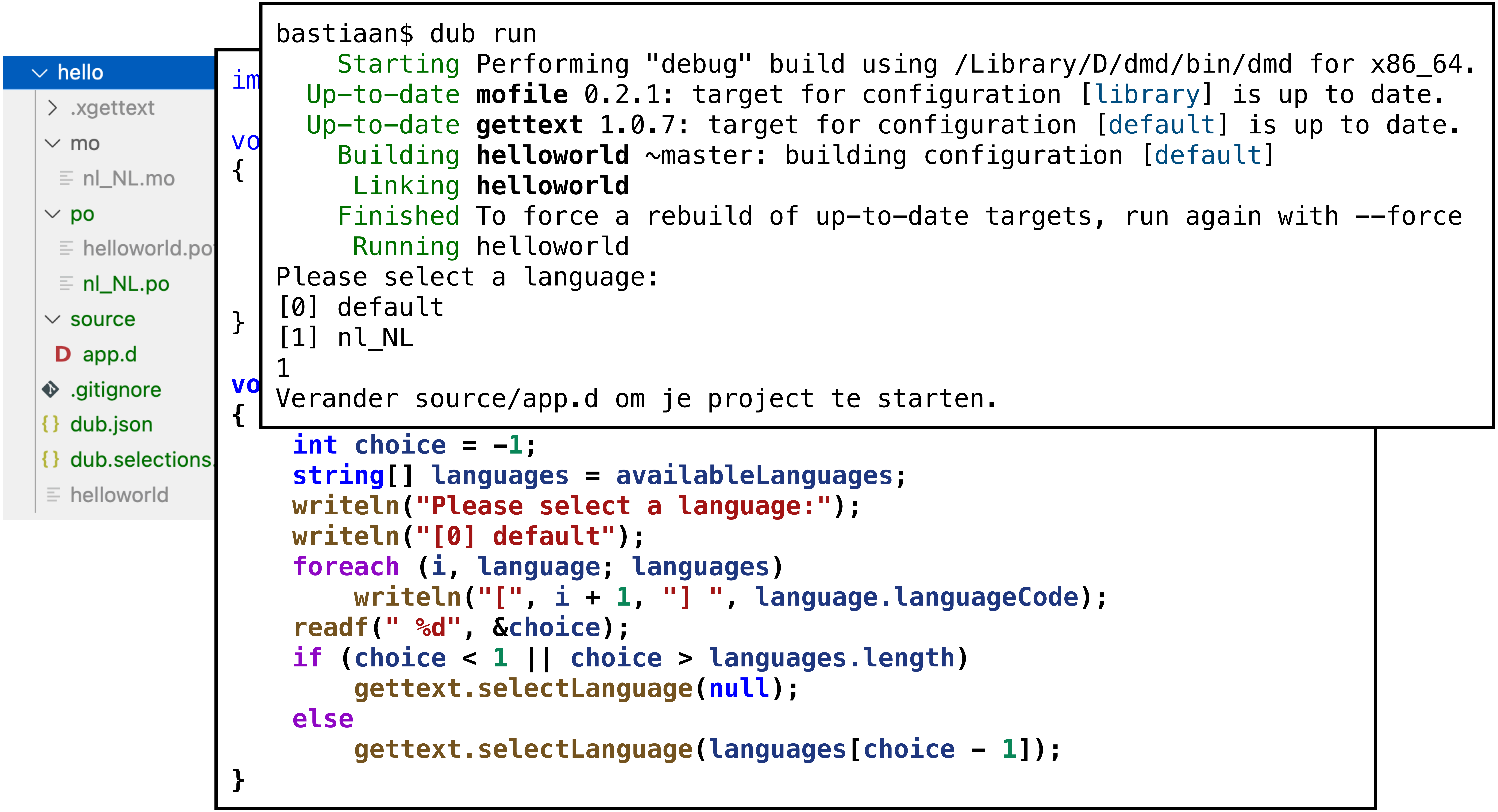


```d
import std.stdio, gettext;

void main()
{
    mixin(gettext.main);

    selectLanguage;
    writeln(tr!"Edit source/app.d to start your project.");
}

void selectLanguage()
{
    int choice = -1;
    string[] languages = availableLanguages;
    writeln("Please select a language:");
    writeln("[0] default");
    foreach (i, language; languages)
        writeln("[", i + 1, "] ", language.languageCode);
    readf(" %d", &choice);
    if (choice < 1 || choice > languages.length)
        gettext.selectLanguage(null);
    else
        gettext.selectLanguage(languages[choice - 1]);
}
```



```
bastiaan$ dub run
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
  Up-to-date gettext 1.0.7: target for configuration [default] is up to date.
    Building helloworld ~master: building configuration [default]
     Linking helloworld
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running helloworld
Please select a language:
[0] default
[1] nl_NL
1
Verander source/app.d om je project te starten.
```

```
im

vo
{



}

vo
{
    int choice = -1;
    string[] languages = availableLanguages;
    writeln("Please select a language:");
    writeln("[0] default");
    foreach (i, language; languages)
        writeln("[", i + 1, "] ", language.languageCode);
    readf(" %d", &choice);
    if (choice < 1 || choice > languages.length)
        gettext.selectLanguage(null);
    else
        gettext.selectLanguage(languages[choice - 1]);
}
```



```
bastiaan$ dub run
    Starting Performing "debug" build using /Library/D/dmd/bin/dmd for x86_64.
  Up-to-date mofile 0.2.1: target for configuration [library] is up to date.
  Up-to-date gettext 1.0.7: target for configuration [default] is up to date.
    Building helloworld ~master: building configuration [default]
     Linking helloworld
    Finished To force a rebuild of up-to-date targets, run again with --force
     Running helloworld
Please select a language:
[0] default
[1] nl_NL
1
Verander source/app.d om je project te starten.
```

```
im

vo
{



}

vo
{
    int choice = -1;
    string[] languages = availableLanguages;
    writeln("Please select a language:");
    writeln("[0] default");
    foreach (i, language; languages)
        writeln("[", i + 1, "] ", language.languageCode);
    readf(" %d", &choice);
    if (choice < 1 || choice > languages.length)
        gettext.selectLanguage(null);
    else
        gettext.selectLanguage(languages[choice - 1]);
}
```

uk_UA.po - Poedit
File
Edit
View
Translation
Go
Help
Open
Save
Validate
Statistics
Pre-translate
Update from code
Upgrade to Pro
Suggestions
Terminology
Add comment

Source text — English	Translation — Ukrainian (Ukraine)
Selected language: %s	Вибрана мова: %s
Identical strings share their translation!	Ідентичні рядки діляться своїм перекладом!
Hello! My name is %s.	Привіт! Мене звати %s.
Never used, but nevertheless translated!	Ніколи не використовувався, але все ж перекладений!
I'm counting one apple.	Я рахую %d яблуко.

Я рахую %d яблуко.
Ctrl+1 • 100% ✓

Я рахую одне яблуко.
Ctrl+2 • 100%

Я рахую одне яблуко.
Ctrl+3 • 95% • translated by DeepL

PRO 8 out of 10 online suggestions left.
Remove this limitation

Source text C format 18 | 23

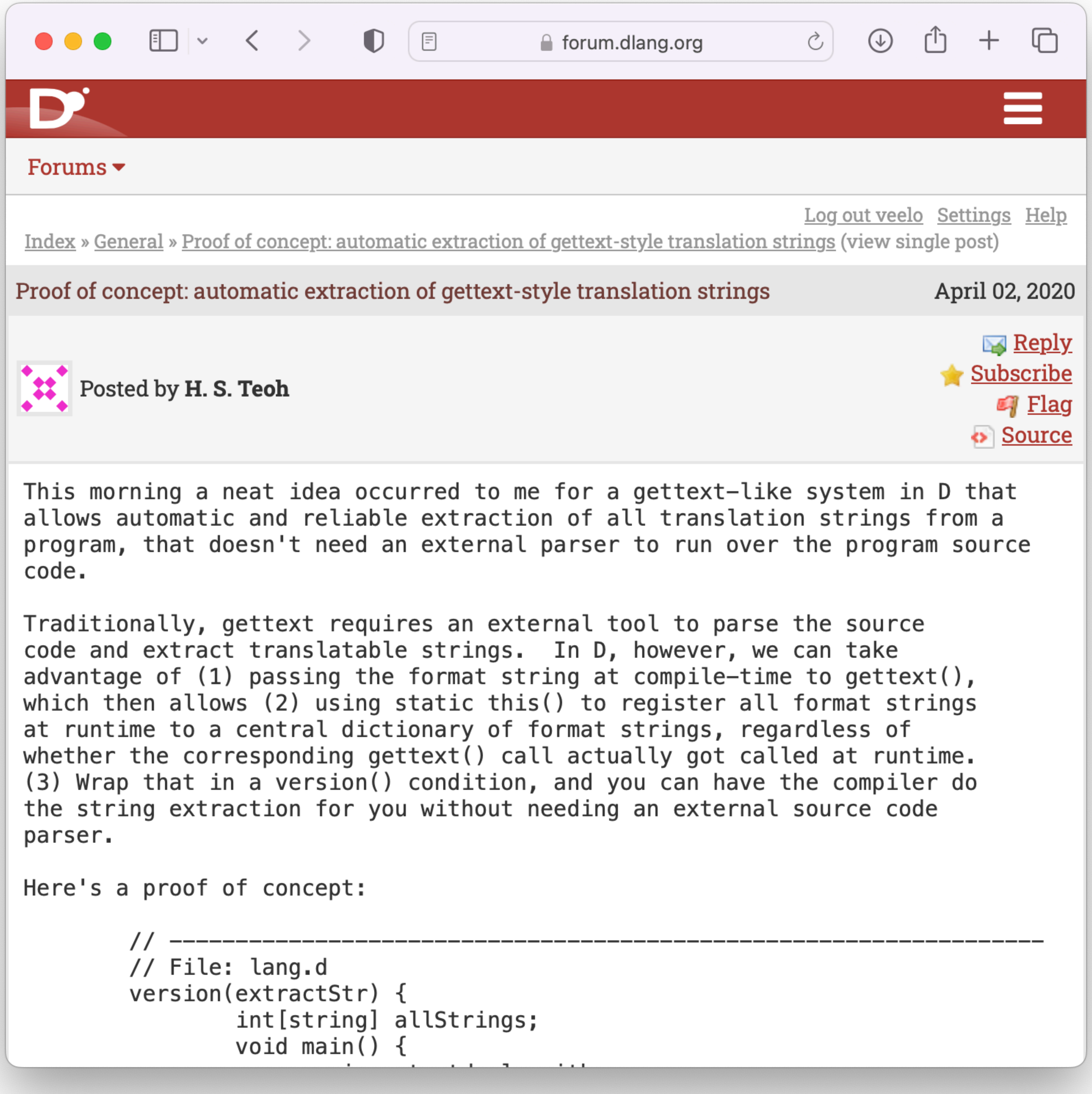
Singular
I'm counting one apple.
Plural
I'm counting %d apples.

Translation Needs work ○

n → 1, 21, 31, 41... | n → 2, 3, 4, 22... | n → 0, 5, 6, 7...

Я рахую %d яблуко.

Translated: 5 of 5 (100 %)



Index » General » Proof of concept: automatic extraction of gettext-style translation strings (view single post)

Proof of concept: automatic extraction of gettext-style translation strings April 02, 2020

Posted by **H. S. Teoh**

```
This morning a neat idea occurred to me for a gettext-like system in D that
allows automatic and reliable extraction of all translation strings from a
program, that doesn't need an external parser to run over the program source
code.

Traditionally, gettext requires an external tool to parse the source
code and extract translatable strings.  In D, however, we can take
advantage of (1) passing the format string at compile-time to gettext(),
which then allows (2) using static this() to register all format strings
at runtime to a central dictionary of format strings, regardless of
whether the corresponding gettext() call actually got called at runtime.
(3) Wrap that in a version() condition, and you can have the compiler do
the string extraction for you without needing an external source code
parser.

Here's a proof of concept:

        // ------------------------------------------------------------
        // File: lang.d
        version(extractStr) {
                int[string] allStrings;
                void main() {
```

https://forum.dlang.org/post/mailman.2526.1585832475.31109.digitalmars-d@puremagic.com

```d
// File: lang.d
version(extractStr) {
    int[string] allStrings;
    void main() {
        import std.algorithm;
        import std.stdio;
        auto s = allStrings.keys;
        s.sort();
        writefln("string[string] dict = [\n%(\t%s: \"\",\n%|%)];", s);
    }
}

template gettext(string fmt, Args...)
{
    version(extractStr)
    static this() {
        allStrings[fmt]++;
    }
    string gettext(Args args) {
        import std.format;
        return format(fmt, args);
    }
}
```

https://forum.dlang.org/post/mailman.2526.1585832475.31109.digitalmars-d@puremagic.com

```d
// File: lang.d
version(extractStr) {
    int[string] allStrings;
    void main() {
        import std.algorithm;
        import std.stdio;
        auto s = allStrings.ke
        s.sort();
        writefln("string[string] dict = [\n%(\t%s: \"\",\n%|%)];", s);
    }
}

template gettext(string fmt, Ar
{
    version(extractStr)
    static this() {
        allStrings[fmt]++;
    }
    string gettext(Args args) {
        import s
        return f
    }
}
```

```d
// File: main.d
import mod1, mod2;

version(extractStr) {} else
void main() {
    auto names = [ "Joe", "Schmoe", "Jane", "Doe" ];
    foreach (i; 0 .. names.length) {
        fun1(names[i]);
        fun2(5 + cast(int)i*10);
    }
}
```

```d
// File: mod1.d
import std.stdio;
import lang;

void fun1(string name) {
    writeln(gettext!"Hello! My name is %s."(name));
}
```

```d
// File: mod2.d
import std.stdio;
import lang;

void fun2(int num) {
    writeln(gettext!"I'm counting %d apples."(num));
}

void fun3() {
    writeln(gettext!"Never called, but nevertheless registered!");
}
```

https://forum.dlang.org/post/mailman.2526.1585832475.31109.digitalmars-d@puremagic.com

```
// File: lang.d
version(extractStr) {
    int[string] allStrings;
    void main() {
        import std.algorithm;
        import std.stdio;
        auto s = allStrings.ke
        s.sort();
        writefln("string[string] dict = [\n%(\t%s: \"\",\n%|%)];", s);
    }
}

template gettext(string fmt, Ar
{
    version(extractStr)
    static this() {
        allStrings[fmt]++;
    }
    string gettext(Args args) {
        import s
        return f
    }
}
```

```
// File: main.d
import mod1, mod2;

version(extractStr) {} else
void main() {
    auto names = [ "Joe", "Schmoe", "Jane", "Doe" ];
    foreach (i; 0 .. names.length) {
        fun1(names[i]);
        fun2(5 + cast(int)i*10);
    }
}
```

```
// File: mod1.d
import std.stdio;
import lang;

void fun1(string name) {
    writeln(gettext!"Hello! My name is %s."(name));
}
```

```
// File: mod2.d
import std.stdio;
import lang;

void fun2(int num) {
    writeln(gettext!"I'm counting %d apples."(num));
}

void fun3() {
    writeln(gettext!"Never called, but nevertheless registered!");
}
```

https://forum.dlang.org/post/mailman.2526.1585832475.31109.digitalmars-d@puremagic.com

```d
// File: lang.d
version(extractStr) {
    int[string] allStrings;
    void main() {
        import std.algorithm;
        import std.stdio;
        auto s = allStrings.ke
        s.sort();
        writefln("string[strin
    }
}

template gettext(string
{
    version(extractStr)
    static this() {
        allStrings[fmt]++;
    }
    string gettext(Args args) {
        import s
        return f
    }
}
```

```d
// File: main.d
import mod1, mod2;

version(extractStr) {} else
void main() {
    auto names = [ "Joe", "Schmoe", "Jane", "Doe" ];
    foreach (i; 0 .. names.length) {
        fun1(names[i]);
        fun2(5 + cast(int)i*10);
    }
}
```

```d
// File: mod1.d


void fun1(string name) {
    writeln(gettext!"Hello! My name is %s."(name));
}
```

```
dmd -i -version=extractStr -run main.d
```

```d
// File: mod2.d
import std.stdio;
import lang;

void fun2(int num) {
    writeln(gettext!"I'm counting %d apples."(num));
}

void fun3() {
    writeln(gettext!"Never called, but nevertheless registered!");
}
```

https://forum.dlang.org/post/mailman.2526.1585832475.31109.digitalmars-d@puremagic.com

```
// File: main.d
import mod1, mod2;

version(extractStr) {} else
void main() {
    auto names = [ "Joe", "Schmoe", "Jane", "Doe" ];
    foreach (i; 0 .. names.length) {
        fun1(names[i]);
        fun2(5 + cast(int)i*10);
    }
}
```

```
// File: lang.d
version(extractStr) {
    int[string] allStrings;
    void main() {
        import std.algorithm;
        import std.stdio;
        auto s = allStrings.ke
        s.sort();
        writefln("string[strin
    }
}

template gettext(string
{
    version(extractStr)
    static this() {
        allStrings[fmt]
    }
    string gettext(Args
        import s
        return f
    }
}
```

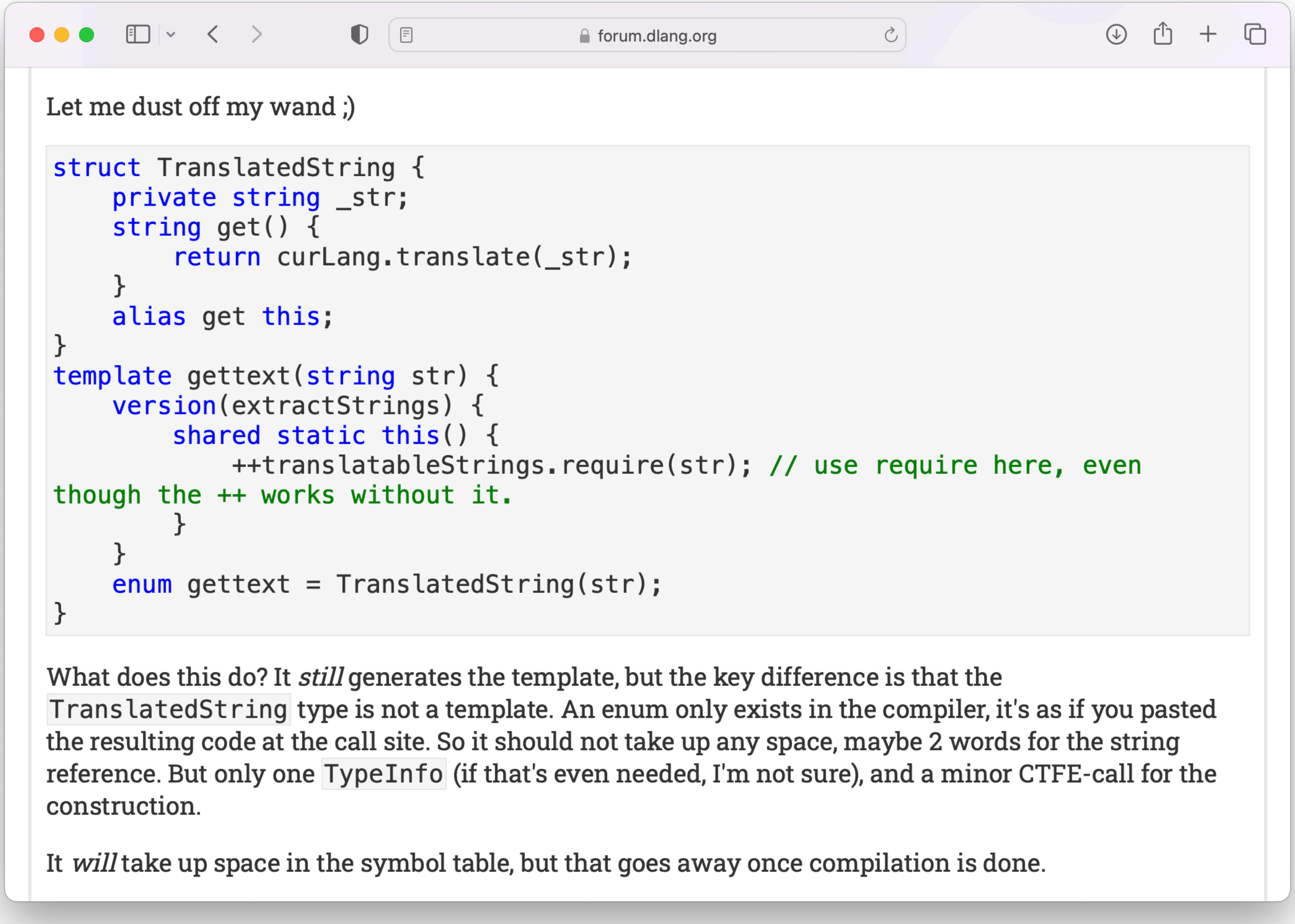
```
// File: mod1.d
```

```
dmd -i -version=extractStr -run main.d
```

```
string[string] dict = [
    "Hello! My name is %s.": "",
    "I'm counting %d apples.": "",
    "Never called, but nevertheless registered!": "",
];
```

```
// File
import std.stdio;
import lang;

void fun2(int num) {
    writeln(gettext!"I'm counting %d apples."(num));
}

void fun3() {
    writeln(gettext!"Never called, but nevertheless registered!");
}
```

https://forum.dlang.org/post/mailman.2526.1585832475.31109.digitalmars-d@puremagic.com



Let me dust off my wand ;)

```
struct TranslatedString {
    private string _str;
    string get() {
        return curLang.translate(_str);
    }
    alias get this;
}
template gettext(string str) {
    version(extractStrings) {
        shared static this() {
            ++translatableStrings.require(str); // use require here, even
though the ++ works without it.
        }
    }
    enum gettext = TranslatedString(str);
}
```

What does this do? It *still* generates the template, but the key difference is that the `TranslatedString` type is not a template. An enum only exists in the compiler, it's as if you pasted the resulting code at the call site. So it should not take up any space, maybe 2 words for the string reference. But only one `TypeInfo` (if that's even needed, I'm not sure), and a minor CTFE-call for the construction.

It *will* take up space in the symbol table, but that goes away once compilation is done.

https://forum.dlang.org/post/t8pqvg$20r0$1@digitalmars.com

Conclusions

Conclusions

- Minimize obfuscation, minimally invasive.

Conclusions

- Minimize obfuscation, minimally invasive.
- Functionally on par with GNU Gettext.

Conclusions

- Minimize obfuscation, minimally invasive.
- Functionally on par with GNU Gettext.
- Low barrier for usage (integrates well in existing Dub project).

Conclusions

- Minimize obfuscation, minimally invasive.
- Functionally on par with GNU Gettext.
- Low barrier for usage (integrates well in existing Dub project).
- Translator friendly.

Conclusions

- Minimize obfuscation, minimally invasive.
- Functionally on par with GNU Gettext.
- Low barrier for usage (integrates well in existing Dub project).
- Translator friendly.

Use it & report any issues!