FCH | Facultad de Ciencias Humanas
ACTIVIDAD DE EXTENSIÓN

UNIVERSIDAD DEL ESTE
LA PLATA

DIPLOMATURA EN SOFTWARE LIBRE

CURSO: Programación II C, C++ / Python

Examen (primera parte “C”)

Institución: Universidad del Este - La Plata - Buenos Aires – Argentina
Período: 2° CUATRIMESTRE de 2016
Profesor coordinador: Mg. Lic. Mariano Reingart

Observaciones Generales

- La entrega se realizará por el campus virtual de la universidad, preferentemente antes del 1 de Noviembre de 2016.
- El trabajo debe ser original, individual y personal, ante cualquier tipo de plagio se invalida el examen y se comunicará la situación a las autoridades de la universidad para que tomen las medidas correspondientes.
- No se aceptarán escritos sin referencias a fuentes relevantes, confiables y verificables. En especial, utilizar Wikipedia o sitios similares sólo como consulta preliminar[1]. Limite de longitud: ver ODT (tamaño máximo: 1MB).
- Para aprobar el examen se debe alcanzar una calificación de 70 puntos. La entrega fuera de fecha quedará a criterio de los docentes, y debe estar debidamente justificada (tanto en las cuestiones personales como en la calidad de los contenidos), o se descontarán puntos por retraso.

Guía de Examen

Análisis de Proyecto en C (50 puntos)

Para el examen se propone trabajar sobre el proyecto GNU Hello[2] (simple e introductorio, desarrollado en el lenguaje de programación C), que permite imprimir un saludo.
Los estudiantes más avanzado que lo deseen pueden elegir un paquete más complejo de la Free Software Foundation: https://www.gnu.org/software/ (de características similares)

Por ejemplo, dentro del paquete sugerido está el programa hello.c (ver anexo) que imprime el tradicional “Hola Mundo”. Según su documentación de GNU Hello, la salida básica (sin traducir) al ejecutarlo es:

1 http://isites.harvard.edu/icb/icb.do?keyword=k70847&pageid=icb.page346376
2 https://www.gnu.org/software/hello/

```
Hello, world!
```

La variante con la opción --traditional:

```
hello, world
```

La otra variante con la opción --greeting=hi :

```
hi
```

Cada estudiante debe preparar un escrito siguiendo los siguientes aspectos (10 puntos c/u):

1. Introducción
   a) Historia y Motivación
   b) Funcionalidades Principales
   c) Desarrolladores
   d) Esquema de Licenciamiento
2. Instalación
   a) Preparación del entorno de desarrollo
   b) Descarga del código fuente, configuración y compilación
   c) Pruebas básicas (correr el programa según lo indica su ayuda)
   d) Pruebas automatizadas (correr los scripts de tests automáticos)
3. Repositorio:
   a) Linea de tiempo (señalando hitos, versiones y cambios principales)
   b) Resumen de issues, bugs, conjuntos de cambios
   c) VCS y ejemplos de uso (clone, pull, update, etc.)
   d) Comparación entre versiones (diff)
4. Explicación del código fuente
   a) Señalar las bibliotecas utilizadas
   b) Indicar el funcionamiento interno del programa (linea por línea)
   c) Identificar la aplicación del Estándar de codificación (estilo) y Prácticas
   d) Resumir el funcionamiento del Mecanismo de Traducción
5. Depuración
   a) Inicio de sesión de depuración (gdb o visual)
   b) Establecer un punto de interrupción (breakpoint)
   c) Correr el programa hasta el punto de interrupción
   d) Inspeccionar las variables

Desarrollo de Homólogo en Python (50 puntos)

Se solicita a cada estudiante implementar un programa similar al GNU Hello, pero en lenguaje de programación Python.

El programa debe ser equivalente y compatible tanto en el funcionamiento (parámetros, entradas y salidas, pruebas) como en poder utilizar los mismos archivos de internacionalización (traducciones).

Debe estar suficientemente comentado (explicando similitudes y diferencias con la versión en C), incluir pruebas unitarias (unittests) y respetar el estilo de codificación de python (PEP8).

Se recomienda utilizar las siguientes bibliotecas estándar de Python ya disponibles para la funcionalidad requerida:

- argparse — Parser for command-line options, arguments and sub-commands
- gettext — Multilingual internationalization services
- unittest — Unit testing framework

Criterios de aprobación: se considerará la "correctitud" (efectividad, terminación, etc.), claridad, diversidad, originalidad y creatividad del proyecto desarrollado, y en especial, de los programas automatizados (por ej., aquellos que usen varios resultados esperados aleatorios; que no comprueben textualmente la salida de los programas; que utilicen correctamente de archivos/flujos E/S, tipos y estructuras de datos y módulos de la biblioteca estándar; que tengan flexibilidad para ser adaptables a otras situaciones; etc.)

En búsqueda de crear conocimiento relevante, libre y abierto según los objetivos de esta diplomatura, el documento, las actividades para Moodle y contenidos relacionados deberán ser licenciado libremente bajo términos legales simples como Creative Commons Atribución-CompartirIgual 2.5 Argentina[3] (CC-BY-SA 2.5 AR o similar) para su eventual publicación.

Los programas desarrollados deberán especificar la Licencia Pública GNU versión 3.0[4] o superior (GPL v3.0+), misma que el proyecto original[5]. El código fuente debe incluirse en el documento (debidamente formateado y explicado, con capturas de pantalla de ejemplos de uso), y además ser subido con anterioridad al repositorio en GitHub: https://github.com/UniversidadDelEste/hello en una carpeta para cada estudiante (nombre/apellido o seudónimo identificable).

3 https://creativecommons.org/licenses/by-sa/2.5/ar/
4 http://www.gnu.org/licenses/licenses.es.html
5 https://docs.moodle.org/dev/License

Código Fuente	Comentarios / Análisis

```
/* hello.c -- print a greeting message and exit.

   Copyright 1992, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2005,
   2006, 2007, 2008, 2010, 2011, 2013, 2014, 2015 Free Software
   Foundation, Inc.

   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.  */

#include <config.h>
#include "system.h"
#include "errno.h"
#include "error.h"
#include "progname.h"
#include "xalloc.h"

/* Forward declarations.  */
static void print_help (FILE *out);
static void print_version (void);

int
main (int argc, char *argv[])
{
  int optc;
  int lose = 0;
  const char *greeting_msg;
  wchar_t *mb_greeting;
  size_t len;

  enum {
    OPT_HELP = CHAR_MAX + 1,
    OPT_VERSION
  };
  static const struct option longopts[] = {
    {"greeting", required_argument, NULL, 'g'},
    {"traditional", no_argument, NULL, 't'},
    {"help", no_argument, NULL, OPT_HELP},
    {"version", no_argument, NULL, OPT_VERSION},
    {NULL, 0, NULL, 0}
  };

  set_program_name (argv[0]);

  /* Set locale via LC_ALL.  */
  setlocale (LC_ALL, "");

#if ENABLE_NLS
  /* Set the text message domain.  */
  bindtextdomain (PACKAGE, LOCALEDIR);
  textdomain (PACKAGE);
#endif

  /* Having initialized gettext, get the default message. */
  greeting_msg = _("Hello, world!");

  /* Even exiting has subtleties.  On exit, if any writes failed, change
     the exit status.  The /dev/full device on GNU/Linux can be used for
     testing; for instance, hello >/dev/full should exit unsuccessfully.
     This is implemented in the Gnulib module "closeout".  */
  atexit (close_stdout);

  while ((optc = getopt_long (argc, argv, "g:t", longopts, NULL)) != -1)
    switch (optc)
      {
          /* --help and --version exit immediately, per GNU coding standards.  */
      case OPT_VERSION:
          print_version ();
          exit (EXIT_SUCCESS);
          break;
      case 'g':
          greeting_msg = optarg;
          break;
      case OPT_HELP:
          print_help (stdout);
          exit (EXIT_SUCCESS);
          break;
      case 't':
          greeting_msg = _("hello, world");
          break;
      default:
          lose = 1;
          break;
```

```
      }

  if (lose || optind < argc)
    {
      /* Print error message and exit.  */
      error (0, 0, "%s: %s", _("extra operand"), argv[optind]);
      print_help (stderr);
    }

  len = mbsrtowcs(NULL, &greeting_msg, 0, NULL);
  if (len == (size_t)-1)
    error (EXIT_FAILURE, errno, _("conversion to a multibyte string failed"));
  mb_greeting = xmalloc((len + 1) * sizeof(wchar_t));
  mbsrtowcs(mb_greeting, &greeting_msg, len + 1, NULL);

  /* Print greeting message and exit. */
  wprintf (L"%ls\n", mb_greeting);
  free(mb_greeting);

  exit (EXIT_SUCCESS);
}

/* Print help info.  This long message is split into
   several pieces to help translators be able to align different
   blocks and identify the various pieces.  */

static void
print_help (FILE *out)
{
  const char *lc_messages = setlocale (LC_MESSAGES, NULL);
  /* TRANSLATORS: --help output 1 (synopsis)
     no-wrap */
  fprintf (out, _("Usage: %s [OPTION]...\n"), program_name);
  /* TRANSLATORS: --help output 2 (brief description)
     no-wrap */
  fputs (_("Print a friendly, customizable greeting.\n"), out);
  fputs ("\n", out);
  /* TRANSLATORS: --help output 3: options
     no-wrap */
  fputs (_("  -t, --traditional       use traditional greeting\n"), out);
  fputs (_("  -g, --greeting=TEXT     use TEXT as the greeting message\n"), out);
  fputs ("\n", out);
  fputs (_("      --help     display this help and exit\n"), out);
  fputs (_("      --version  output version information and exit\n"), out);
  fputs ("\n", out);
  /* TRANSLATORS: --help output 4+ (reports)
     TRANSLATORS: the placeholder indicates the bug-reporting address
     for this application.
     no-wrap */
  fprintf (out, _("Report bugs to: %s\n"), PACKAGE_BUGREPORT);
  /* Don't output this redundant message for English locales.
     Note we still output for 'C' so that it gets included in the man page.  */
  if (lc_messages && STRNCMP_LIT (lc_messages, "en_"))
    {
      /* TRANSLATORS: Replace LANG_CODE in this URL with your language code
           <http://translationproject.org/team/LANG_CODE.html> to form one of
           the URLs at http://translationproject.org/team/.  Otherwise, replace
           the entire URL with your translation team's email address.  */
       fprintf (out, _("Report %s translation bugs to "
                       "<http://translationproject.org/team/>\n"), PACKAGE_NAME);
    }
#ifdef PACKAGE_PACKAGER_BUG_REPORTS
  fprintf (out, _("Report %s bugs to: %s\n"), PACKAGE_PACKAGER,
           PACKAGE_PACKAGER_BUG_REPORTS);
#endif
#ifdef PACKAGE_URL
  fprintf (out, _("%s home page: <%s>\n"), PACKAGE_NAME, PACKAGE_URL);
#else
  fprintf (out, _("%s home page: <http://www.gnu.org/software/%s/>\n"),
           PACKAGE_NAME, PACKAGE);
#endif
  fputs (_("General help using GNU software: <http://www.gnu.org/gethelp/>\n"),
         out);
  exit(out == stderr ? EXIT_FAILURE : EXIT_SUCCESS);
}

/* Print version and copyright information.  */

static void
print_version (void)
{
  printf ("%s (%s) %s\n", PACKAGE, PACKAGE_NAME, VERSION);
  /* xgettext: no-wrap */
  puts ("");

  /* It is important to separate the year from the rest of the message,
     as done here, to avoid having to retranslate the message when a new
     year comes around.  */
  printf (_("\
Copyright (C) %d Free Software Foundation, Inc.\n\
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>\n\
This is free software: you are free to change and redistribute it.\n\
There is NO WARRANTY, to the extent permitted by law.\n"), COPYRIGHT_YEAR);
}
```