DIPLOMATURA EN SOFTWARE LIBRE

Curso: Programación II C C++ Python

Planificación inicial y Temario detallado de estudio – Versión Preliminar tentativa
Institución: Universidad del Este - La Plata - Buenos Aires – Argentina
Período: 2º CUATRIMESTRE de 2015
Docente a cargo: Mg. Lic. Mariano Reingart
Horas semanales: 8

PRESENTACIÓN GENERAL

El Proyecto GNU fue iniciado en 1984 para crear un sistema operativo basado en software libre similar a UNIX y, así promover la libertad y la cooperación entre usuarios de ordenadores y programadores.
Cualquier sistema operativo basado en UNIX necesita un compilador de C, y no había compiladores libres en ese momento, el Proyecto GNU debía desarrollar uno desde cero.
Este trabajo fue financiado por donaciones de individuos y compañías a través de la Free Software Foundation, una organización sin ánimo de lucro destinada a dar soporte al trabajo del Proyecto GNU.
La primera entrega de GCC fue hecha en 1987. Esto fue un significativo progreso, siendo el primer compilador portable para optimizar ANSI C liberado como software libre. Desde este momento GCC ha llegado a ser uno de las más importantes herramientas en el desarrollo de software libre.
Un avance importante en el compilador llega con la serie 2.0 en 1992, que añade la capacidad de compilar C++.
A través del tiempo GCC ha sido extendido para dar soporte a muchos lenguajes adicionales, incluyendo Fortran, ADA, Java y Objective-C. El acrónimo GCC es ahora usado para referir al “GNU Compiler Collection” (Colección de Compiladores de GNU). Su desarrollo está guiado por el GCC Steering Committee, un grupo compuesto de representantes de las comunidades de usuarios/as de GCC en la industria, la investigación y la academia.
Una Introducción a GCC. para los Compiladores de GNU gcc y g++ Revisado y actualizado Brian Gough Prefacio por Richard M. Stallman . 2005

OBJETIVOS

Este curso tiene como objetivo fundamental brindar los conocimientos, habilidades y el entrenamiento intermedio para que el alumno pueda desarrollar soluciones mediante la programación utilizando estructuras de datos y control más avanzadas. Esto implica:

- Facilitar a los alumnos los conceptos teóricos necesarios para comprender la importancia de la a programación de sistemas en la actualidad
- Facilitar el desarrollo de técnicas orientadas a la optimización de programas
- Facilitar el entendimiento, el desarrollo y la comprensión de la ejecución de programas de mediana y alta complejidad.

Por ello se espera que el alumno, al aprobar el curso, pueda:

- Aplicar diferentes estructuras de datos y de control adecuándolas a cada problema particular.
- Diseñar algoritmos de mediana complejidad.

• Producir de manera óptima programas adecuando los tipos de datos, aplicando los criterios de programación modular y los de refinamiento sucesivo.
• Codificar los algoritmos en un lenguaje de programación estructurado y orientado a objetos como C, C++ y Python.

Este curso busca también ser una preparación introductoria a la programación avanzada utilizada en proyectos de software libre como el kernel de Linux, el lenguaje de Programación Python o la base de datos PostgreSQL (escritas en C), como a su ves módulos y extensiones en C++ para los lenguajes de programación de alto nivel.
Dado que se profundizan conceptos generales, es recomendable haber cursado anteriormente “Programación I Python”. Dado que varios conceptos se interrelacionan con el curso “Sistemas operativos GNU/Linux”, es recomendable cursarlo en paralelo.

UNIDADES DIDÁCTICAS

Eje temático central: lenguajes estáticos de programación de sistemas C, C++ y dinámicos Python

Unidad 1. Introducción a la programación en C

Un poco de historia de C. Entorno de programación en GNU/C. Estructura de un programa simple. La programación imperativa. Tipos de datos básicos. Asignación y evaluación de expresiones. Instrucciones de selección. Funciones estándar de entrada y salida.

Unidad 2. La programación estructurada

Principios. Instrucciones iterativas. Procesamiento de secuencias de datos. Esquemas algorítmicos: recorrido y búsqueda. Filtros y tuberías. Depuración de programas. Estructuras de datos. Matrices. Estructuras heterogéneas. Tipos de datos abstractos. Tipos enumerados. Ficheros. Principios de la programación modular. Funciones. Macros del preprocesador C.

Unidad 3. Programación avanzada en C. Desarrollo eficiente de aplicaciones

Las variables dinámicas. Los apuntadores. Creación y destrucción. Tipos de datos dinámicos (cadenas de caracteres, . Diseño descendente de programas. Tipos de datos abstractos y funciones asociadas. Ficheros de cabecera. Bibliotecas. Make. Relación con el sistema operativo. Ejecución de funciones del sistema operativo. Gestión de procesos. Hilos. Procesos.

Unidad 4. Programación orientada a objetos en C++

Introducción. De C a C++. El paradigma de programación orientada a objetos. Clases y objetos. Acceso. Constructores y destructores. Organización y uso de bibliotecas en C]++. Diseño de programas orientados a objetos. Conceptos: homonimia, herencia simple, polimorfismo.

Unidad 5. Programación en Python

Introducción. Origen de Python. Características generales de Python. Entorno de desarrollo Python. Diferencias entre C++ y Python. Las clases en Python. Herencia y Polimorfismo. Programación asincrónica y orientada a aspectos. Hilos. Interfaces gráficas.

METODOLOGÍA

Cada alumno trabajará desde su computadora individual comunicados vía Internet por el campus virtual, enfocados en una guía de ejercicios para cada unidad.
El profesor asumirá un método de enseñanza directa ya que conducirá y dictará las pautas a seguir, realizando una exposición del tema a desarrollar (con demostraciones de las actividades prácticas o presentación de temas teóricos), para que luego los alumnos puedan reproducir y analizar los nuevos contenidos a aprender.
Se busca un aprendizaje significativo, favoreciendo que los alumnos relacionen con mayor facilidad los conocimientos aprendidos a lo largo de la diplomatura. Este tipo de aprendizaje es el que hará efectiva la comprensión de los temas, donde habrá un proceso de elaboración de conocimiento de una manera integral.

EVALUACIÓN

Las evaluaciones serán formativas con base constructivista (evitando la mera memorización mecánica o repetitiva), de carácter procesual (teniendo en cuenta los procedimientos realizados y no solo el resultado final), con contenidos teóricos y prácticos, llevadas a cabo principalmente en el campus virtual. El alumno deberá poder contestar preguntas teóricas sobre los contenidos vistos en el curso y desarrollar ejercicios prácticos usando las herramientas utilizadas durante la cursada. Se evaluará de manera continuada, bajo modelos de evaluación tradicional (con varios exámenes parciales domiciliarios por cuatrimestre: actividades -trabajos prácticos- a entregar por el campus virtual) y enfoques alternativos actualizados medida por tecnología
(observaciones de la participación y colaboración activa con criterios concretos). Se apunta a una evaluación sumativa de todos los conceptos, que refiera a los logros y rendimiento del aprendizaje de los alumnos, debiendo poner de manifiesto la internalización de los conceptos abordados.
Cada evaluación parcial será escrita por computadora (con elementos multimedia), en la que el alumno desarrollará los contenidos teórico-prácticos vistos en el curso en base a una guía de examen facilitada por el profesor. El alumno deberá elaborar y entregar en tiempo y forma todas las actividades obligatorias que proponga el equipo docente.

Actividades:

Ver ejercitación en la bibliografía (una actividad de autoevaluación a entregar por unidad). Adicionalmente se contempla la presentación de trabajos prácticos complementario sobre el estudio de un modulo del kernel de GNU/Linux (C), extensión de Python (C++) y aplicación visual en wxPython, a elección del alumno.

Calificación:

La calificación será porcentual (0 a 100), siendo 70 el valor mínimo para aprobar los exámenes, actividades y ejercicios. No se contemplan recuperatorios, por lo que el alumno deberá aprobar al menos 2/3 de las actividades propuestas por el docente.
Para acreditar los conocimientos, el alumno deberá aprobar los exámenes parciales. Una vez que el alumno haya cumplido con la aprobación de los exámenes, se entregará al alumno un certificado emitido por la universidad donde conste que el alumno ha aprobado el curso.

EXPERIENCIA DOCENTE

Mariano Reingart es Licenciado en Sistemas, Magíster en Software Libre (UOC) y actualmente finalizando el Profesorado en Disciplinas Industriales (UTN-INSPT). Es docente en el Instituto Superior Blaise Pascal desde 2009 en materias de 2º y 3º Año (actualmente «Bases de Datos», «Sistemas Operativos», «Interconectividad» y «Práctica Profesional») de las carreras terciarias «Tecnicatura superior en Análisis de Sistemas / Redes Informáticas». Ha trabajado como Analista-Programador Freelance (en varias empresas del sector y actualmente en un emprendimiento propio). En el área del software libre, es miembro de varias asociaciones y grupos de usuarios/desarrolladores, con activa participación en varios proyectos. En especial, es miembro de la Python Software Foundation, habiendo disertado en varias conferencias y colaborado en múltiples proyectos de software libre que utilizan este lenguaje de programación. Ha ha desarrollado módulos y controladores de comunicación para el núcleo del Linux (ad-hoc), y participado como estudiante en un proyecto de para el programa “Google Summer of Code 2014”, desarrollando en C++ un port de la interfaz gráfica Qt para wxWidgets / wxPython (tanto bajo GNU/Linux como Android).

BIBLIOGRAFÍA GENERAL

Introducción al desarrollo del software

Josep Anton Pérez López, Lluís Ribas i Xirgo, Primera edición: marzo 2004 © Fundació per a la Universitat Oberta de Catalunya
http://ocw.uoc.edu/computer-science-technology-and-multimedia/introduction-to-softwaredevelopment/introduction-to-software-development/XP06_M2110_01498.pdf