Documentación Proyecto #2

Juan Esteban Navarro Camacho,Esteban Ferarios, Santiago Gamboa Ramírez
santigr17@gmail.com
ferarios95@gmail.com
jnavcamacho@gmail.com
Instituto Tecnológico de Costa Rica
Ingeniería en Computadores
Principios de Sistemas Operativos

***Resumen*—El siguiente documento presenta el diseño y los resultados obtenidos de la implementación de una biblioteca de procesos livianos inspirada en PThread con el fin de realizar diversos mecanismos de calendarización los cuales serán visualizados en una animación realizada en Allegro 5, el cual es un generador de interfaz gráfica en C.**

***Palabras clave*—Allegro, C, GitHub, LPThread, Procesos Livianos, Planificador, PThread, Valgrind**

I. Introducción

Es los sistemas operativos actuales, es necesario tomar en cuenta que estos sistemas presentan al usuario mecanismos interactivos para realizar las diversas tareas asociadas a las necesidades del mismo. Los hilos al igual que los procesos, promueven la condición de multiprogramación. Estos permiten así atender operaciones de entradas y salidas, cálculos, interrupciones, etc de forma cuaci-paralela.
Los hilos actuales representan procesos, (livianos), los cuales comparten el mismo espacio de direcciones y todos los datos del proceso padre creador, requieren de menor inversión de recursos para crearlos o destruirlos y permiten a una misma aplicación en ejecución por poder atender la interacción con el usuario, realizar escrituras a memoria secundaria y atender señales del sistema[TAN09].

Además de facilitar la condición de multiprogramación es importante considerar que esto implica que el sistema debe de contar con mecanismos para controlar de forma programada el momento de ejecución de cada hilo, (planificación), así como el momento y la forma en la que accede y administra recursos compartidos.

I-A. Aspectos importantes sobre los hilos

Cabe destacar que tanto los procesos como los hilos, (procesos livianos) son creados, con la ayuda de una misma llamada al sistema, **clone**, la cual permite la creación tanto de procesos como de hilos, mediante las banderas que así se le suministren. [Ben18]

Algunas características de los hilos son las siguientes:

- Contador de programa, se lleva registro de cual instrucción se debe de ejecutar a continuación.
- Variables de trabajo actuales
- Pila, con el historial de ejecución.
- Los hilos de un mismo proceso pesado comparten el mismo espacio de direcciones.
- No existe protección entre hilos asociados a un mismo proceso.
- Los hilos de un mismo proceso pueden compartir el mismo conjunto de archivos abiertos, procesos hijos, alarmas y señales.
- Los hilos poseen estados: Ejecución, Bloqueado, listo, terminado.
- Los hilos pueden esperar a otros de su misma naturaleza terminen. generalmente esta característica es identificada como la función **thread_join.**
- Los hilos pueden entregar *voluntariamente* la CPU para dejar que otro hilo se ejecute.
- Los hilos permiten que cada proceso tenga su propio algoritmo de planificación de hilos personalizados.
- Hay dos formas de implementar un paquete de hilos, mediante el espacio de usuario o mediante el espacio del kernel.

I-B. Hilos bajo el estándar de POSIX

Es un estándar definido por la IEEE el cual se identifica como 1003.1c A continuación se describen algunas de las característica.

- Identificador
- Conjunto de registro
- Atributos: Tamaño de la pila, parámetros de planificación

Cuadro I
Algunas de las llamadas a funciones de Pthreads.

Llamada del hilo	Descripción
Pthread_create	Crea un nuevo hilo
Pthread_exit	Termina el hilo llamador
Pthread_join	Espera a que un hilo específico termine
Pthread_yield	Libera la CPU para dejar que otro hilo se ejecute
Pthread_attr_init	Crea e inicializa la estructura de atributos de un hilo
Pthread_attr_destroy	Elimina la estructura de atributos de un hilo

I-C. Principales bibliotecas y funciones utilizadas para la implementación de la biblioteca de hilos y la interfaz gráfica

I-C1. ***Hilos, procesos livianos****:*

- **atexit(...args)** Registra la función dada que se llamará en la finalización normal del proceso, ya sea a través de la salida o mediante el retorno desde el programa principal [Ker20e].
- **clone(...args)** Esta llamada al sistema cra un nuevo proceso, pesado o liviano. Proporciona mayor control sobre qué será compartido del contexto de ejecución entre el proceso que llama a clone así como al proceso creado. [Ker20d].
- **getpid()** Retornar el identificador del propio proceso que invoca la función [Ker20c].
- **getpgrp()** Retorna el identificador del grupo asociado al proceso que invoca la llamada [Gro20].
- **kill(...args)** Esta llamada al sistema se puede usar para enviar cualquier señal a cualquier proceso grupo o proceso[Ker20a].
- **killpg(...)**; Envía una señal a un grupo de procesos [Ker20b].
- **posix_memalign(...)** Esta función asignará bytes de tamaño alineados en un límite especificado por la alineación, y devolverá un puntero a la memoria asignada. El valor de alineación será un múltiplo de una potencia de base 2 del valor de sizeof (void *) [Gro19].
- **sched_yield()** Hace que el hilo de llamada abandone la CPU. los el subproceso se mueve al final de la cola por su prioridad estática y un nuevo hilo llega a ejecutarse [Ker19].
- **waitpid(...args)** Esta función suspenderá la ejecución del hilo que la invoca hasta el estado para uno o más procesos esté disponible [IS19].

I-C2. ***Interfaz gráfica****:* Alegro es una biblioteca multiplataforma para el desarrollo de juegos e interfaz gráfica general, posee capacidades para presentar fuentes de texto, dibujar figuras geométricas y presentar multimedia, así como responder a eventos por ejemplos entradas de periféricos de la computadora [Tea10].

En este documento se encuentran las experiencias obtenidas en la realización del segundo proyecto del curso Principios de Sistema Operativos, en donde mediante una simulación se representarán los siguientes Algoritmos de calendarización en procesos livianos, cuya implementación está inspirada en la biblioteca conocida como PThread y cuya simulación se realizarán en figuras desplazándose sobre estructuras. de una sola vía. Este proyecto aborda los objetivos asociados a implementar los algoritmos de calendarización de procesos livianos así como interactuar con las directivas del procesador.

II. Ambiente de desarrollo

Esta sección se detallan las herramientas utilizadas durante el desarrollo de este proyecto. Cabe destacar que se utiliza GNU-Linux como sistema operativo por defecto para las distribuciones Ubunut 18.04 y Debian 9.

- **Entorno de desarrollo integrado** Se utiliza el entorno de Visual Studio Code, con los los siguientes complementos para garantizar, documentación de código, errores sintácticos y asistencia de código general, además de la integración.
  - *Code Spell Checker*: Esta herramienta permite reducir la incidencia en los errores ortográficos a la hora de establecer documentación interna y definición de mnemónicos el código de C [**Street Side Software**].
  - *GitLens — Git supercharged* Permite la integración de artefactos visuales para identificar en cada línea de código, quién fue la persona responsable, así como una rápida visualización de la versión del código en commits previos dado un branch. Git lens cuenta con herramientas gráficas para visualizar aspectos relacionados con las diferencias entre versiones de archivos, operaciones de merge/rebase y navegación entre branches del repositorio [Amo16].
  - *C/C++ for VS code* Este complemento permite aplicar operaciones de formato para software escrito en C. Valida integridad de tipo en el código escrito. Revisa que cada biblioteca se incluida correctamente y proporciona **code assistance** para una escritura de código más eficiente permitiendo autocompletar las sentencias [Mic16].
- **Comunicación con el equipo** Las principales plataformas para realizar las reuniones grupales son las siguientes:
  - *Zoom* Cuyo principal valor es la posibilidad de poder compartir el escritorio de la computadora de forma completa y la herramienta de dibujo, pues es vital para poder realizar observaciones y anotaciones sobre sobre la interfaz gráfica de la simulación del proyecto permitiendo desarrollar de forma más adecuada las ideas de los miembros, pues dichas ideas tenían un alto componente visual.
  - *Whereby* Es una plataforma web que permite reuniones sin límite de tiempo para cuatro personas, al igual Zoom cuanta con una pizarra virtual integrada en la sala de reuniones para que los miembros desarrollen de forma colaborativa cada una de las ideas o planteamientos del proyecto.
- **Definición y distribución de tareas** Las tareas son definidas mediante el esquema de canvas, tradicional en los proyectos de software, agrupando las tareas con estados en los cuales se indica qué tarea está pendiente, qué tarea se está resolviendo y qué tarea está finalizada. Está herramienta esta integrada por defecto en el sitio oficial de GitHub donde se encuentra el repositorio de versiones del proyecto.
- **Compilación y pruebas del software** Se desarrolla un archivo de script en bash el cual permite instalar cada una de las dependencias para ejecutar el proyecto. Para la compilación del proyecto de forma amigable con el usuario utilizamos archivos Makefiles los cuales contienen la complejidad de los comandos, parámetros y sintaxis para la construcción y ejecución del proyecto.
- **Herramientas de depuración** Valgrind GDB
  - *Valgrind* Es un marco de trabajo que permite

mediante sus módulos operaciones de depuración, análisis de rendimiento y mejoras, entre ellos algunos complementos son los siguientes[Dev20].

- Memcheck está asociado a la detección de errores en memoria.
- Helgrind se vincula a la detección de errores en hilos.
- DRD al igual que Helgrind involucra análisis de detección de errores hilos.

- *GDB* Permite examinar qué está sucediendo con los componentes de un programa durante su ejecución, incluyendo lo que dicho programa estaba haciendo previo a la ocurrencia de un error. GDB permite [dev20].
  - Iniciar el programa, especificando cualquier cosa que pueda afectar su comportamiento.
  - Provocar que el programa se detenga en condiciones específicas.
  - Examinar lo que sucedió cuando el programa se detuvo.
  - Modifique valores en su programa, con el fin de poder experimentar, corrigiendo los efectos de un error y continuar aprendiendo sobre otro.

- Versionamiento del código: Se utiliza Git Hub como plataforma para versionar el código fuente permitiendo a su vez apoyarse de la estrategia de git flow para crear funcionalidades nuevas y propiciar su integración adecuada .

III. Detalles del diseño del programa desarrollado, tanto del software

III-A. Modelo de la implementación de la bibliotecas de hilos Hilo

A continuación se presentan diagramas de componentes con su respectiva descripción de las funciones utilizadas para implementar la biblioteca de hilos LPthread.

III-A1. ***Creación de hilos****:* Se utiliza la llamada a la función clone, suministrando las banderas asociadas a la creación de procesos livianos. En resumen se utilizan: **CLONE_FS** la cual permite clonar el sistema de archivos asociado a la llamada del proceso, **CLONE_FILES** para poder compartir la misma tabla de descriptores del archivo, **CLONE_SIGHAND** tanto el proceso padre como el proceso hijo compartirán la misma tabla de administradores de señales, **CLONE_VM** el proceso padre y el proceso se ejecutarán en el mismo espacio de memoria, **CLONE_PTRACE** si el proceso está siendo rastreado, (trace), así lo será también el proceso hijo.

III-A2. ***Finalización de Hilos****:* En términos de finalización de un hilo se utiliza la función **kill(id, SIGKILL);** en donde se suministra la señal de finalización del proceso liviano.

III-A3. ***Ceder recursos de forma voluntaria****:* Se utilizó la opción **sched_yield** la cual hace que el hilo ceda el uso del recursos (procesador).

III-A4. ***Espera de hilos****:* Mediante el llamado a la función **wait** indicando qué el proceso debe de esperar hasta que el proceso objetivo cambie de estado.

III-B. Modelado de la interfaz gráfica

Algunos componentes de la interfaz gráfica, son estáticos, otros componentes son generados de forma dinámica entre ellos el componente Path que representa un bloque identificado como (2) en la imagen III-D. Los planetas son creados de forma estática y representan el punto de origen en donde los aliens entran y salen.

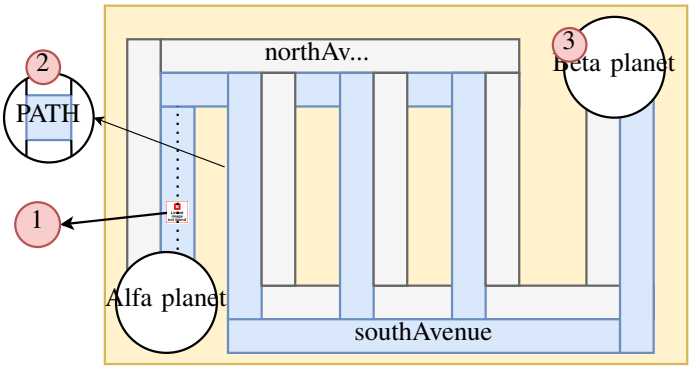

Figura 1. Principales componentes de la interfaz gráfica

III-B1. Tránsito de los aliens en la interfaz gráfica: Los aliens tienen permitido su desplazamiento sólo en las figuras de color azul o gris de la imagen III-D cada color tiene un sentido permitido de desplazamiento. Cada Alien conoce qué ruta tomar para atravesar de un planeta o comunidad a la otra. Esto implica que la abstracción de la ruta se logra agrupando una serie de arreglos lo cuales con la información de cada segmento recto de su trayectoria. Cada segmento recto se identifica como: PATH el cual es un puntero con la información asociada a un arreglo dinámico de bloques de tipo PATH. Cada bloque PATH contiene información de sobre las coordenadas del bloque, si está bloqueado o no y si contiene un Alien en dicha posición.

III-C. Abordaje de planificadores

III-C1. ***Round Robin****:*

- Se ordena una cola de Aliens en FIFO (Ready Queue) que pasan de un estado NEW a READY.
- Si se tiene el recurso(Paso al puente) y el primer alien de la lista cumple los requisitos(el puente lo resiste).
- Este pasara de un estado READY a RUNNING el Alien procede a intentar cruzar el puente.
- (Se remueve de la Ready queue y se agrega en la Running queue).
- Si el recurso se le expropia y el alien aún se encuentra en el puente (Running Queue) se le cambia el estado a WAITING y vuelve a la Ready queue de manera especial.
- La próxima vez que tenga recurso y pase al puente (Running queue) recordara el estado en el que estaba (El Path por el iba).

III-C2. ***Planificación por prioridad****:*

- Se ordena una cola de Aliens en FIFO (Ready Queue) que pasan de un estado NEW a READY

III-C3. El más corto primero:

- Se ordena una cola de Aliens en función del Alien más rápido ya que este terminariía antes en pasar el puente (Ready Queue) que pasan de un estado NEW a READY.
- Si se tiene el recurso(Paso al puente) y el primer alien de la lista cumple los requisitos(el puente lo resiste) este pasara de READY A RUNNING y se movera al puente.

III-C4. Primero en llegar, primero en salir:

- Se ordena una cola de Aliens en FIFO (Ready Queue) que pasan de un estado NEW a READY
- Si se tiene el recurso(Paso al puente) y el primer alien de la lista cumple los requisitos(el puente lo resiste) este pasara de READY A RUNNING y se moverá al puente (Se pasa de Ready queue a Queue).

III-C5. Lotería:

- Se ordena una cola de Aliens en Usando una loteria (Ready Queue) que pasan de un estado NEW a READY.
- Se crea una Lista nueva vacia
- Se crea una copia de la Lista vieja (Ready Queue)
- Algoritmo de loteria: Se toma todos los Aliens en lista vieja y suma sus tiquetes en una variable
- Crea un array de integers del largo de esa suma
- Luego toma cada Alien e itera el número de tiquetes que tiene ,se agrega su ID en el array anterior cada iteración , esto con cada alien de la lista
- Se hace un número aleatorio de 0 a la suma de tiquetes -1
- Se saca el ID del ganador del array con el random
- Se elimina ese ganador de la lista original y se hace push en una lista nueva:
- Se elimina el ganador de la lista vieja y se repite el sorte con la lista vieja de nuevo
- Se repite el proceso hasta tener una lista nueva igual a la del tamaño original.
- Una vez con la lista de ganadores en orden de loteria se devuelve esta.
- El sistema reemplaza la lista original con esta nueva
- Si se tiene el recurso(Paso al puente) y el primer alien de la lista cumple los requisitos(el puente lo resiste) este pasara de READY A RUNNING y se movera al puente (Se pasa de Ready queue a Running Queue).

III-D. Integración de la interfaz gráfica con la simulación

El siguiente esquema representa de forma general la intervención de cada una de las tareas que se realizan mediante la biblioteca implementada de hilos. Es diagrama incluye las etapas de configuración además de la ejecución de los principales escenarios que suceden en la interfaz dadas las combinaciones de planificadores y cruce a través de los puentes.

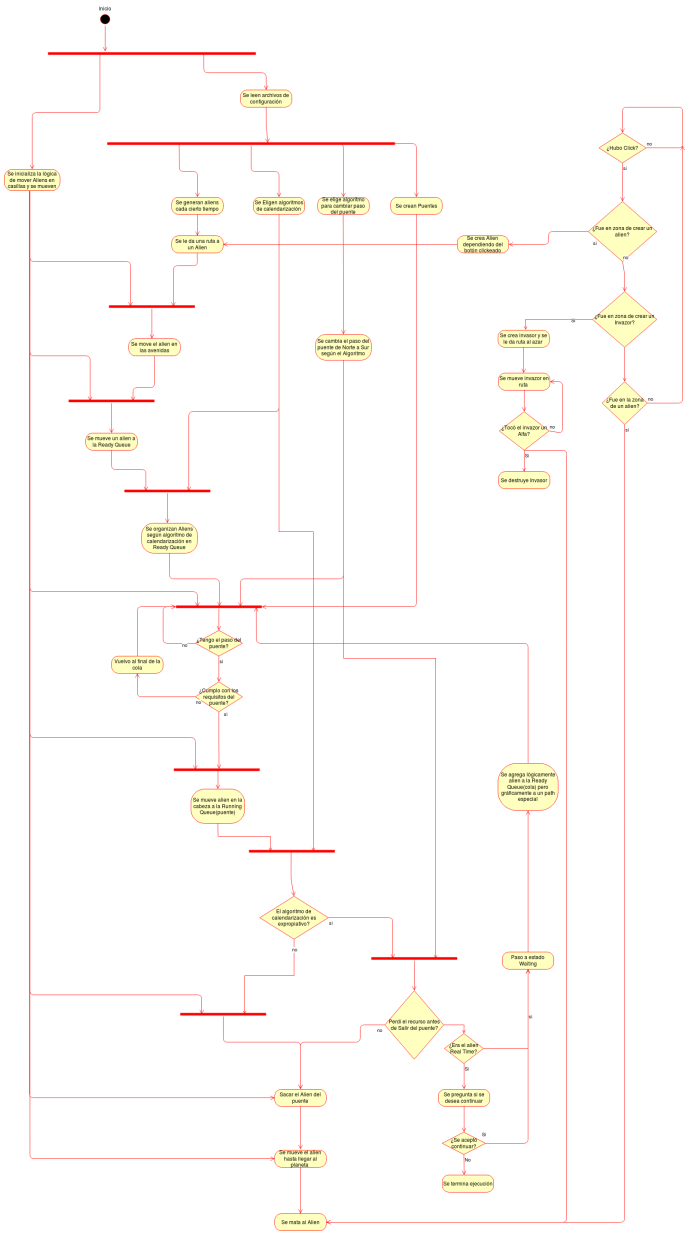

Figura 2. Principales componentes de la interfaz gráfica

IV. MANUAL DEL USUARIO

En esta sección se presenta los pasaos necesario para probar el proyecto de simulación de Aliens.

IV-A. Instalación del proyecto

Posterior a realizar clone del proyecto de GitHub, se procede a ejecutar el script de instalación definido en el archivo install.sh, el cual instala la biblioteca para la creación de Gráficos en C. Es necesario aportar las credenciales de super usuario pues se instalará software.

IV-B. Compilación del programa

En la raíz del proyecto se procede a ingresar a la carpeta GUI en donde se ejecuta el comando make, el cual tiene una tarea definida para la compilación y ejecución del proyecto.

IV-C. Personalizar la simulación

Para establecer los parámetros de la simulación se ingresa desde la raíz del proyecto a la carpeta **/gui/config** en donde están los siguientes archivos:

- alien.conf: Configuración para definir la velocidad base, tiempo en RTOS, media de tiempo creación asociada a la distribución de tiempo de creación de los aliens. Proporción de Aliens basado
- ¡east,mid,west¿_bridge.conf: Cada archivo cuneta con la documentación para definir el tipo de planificador y lel algoritmo de cruce de los puentes incluyendo la longitud, y tiempo para ingreso al puente tanto de la salida norte como sur.

Posterior a la configuración de los planificadores, se procede regresar a la carpeta **gui** en donde con el comando make se ejecutarán de nuevo la aplicación con la nueva configuración establecida.

IV-D. Uso del programa

En la siguiente imagen se describen los botones a los cuales el usuario tiene acceso.

- **1)** Este botón activa el alien invasor, atacando a los aliens alfa, identificados con el color azul. Al destruir un alien alfa el invasor desaparecerá y podrá ser invocado de nuevo. Aparecerá de forma aleatoria ya sea en la avenida norte o sur.
- **2)** Este botón permite crear los aliens normales.
- **3)** Este botón establece que el lugar de salida es el planeta Alfa.
- **4)** Establece el planeta beta para la creación de aliens.
- **5)** Crea un alien de tipo alfa.
- **6)** Crea un alien de tipo de beta.
- **7)** Información de las estadísticas de cada puente.

Figura 3. Interfaz gráfica

V. COEVALUACIÓN

La valoración es de 1 a 10, es decir cada estudiante evaluará a sus otros 3 compañeros de 1 a 10.

Cuadro II
COEVALUACIÓN DE LOS ESTUDIANTES

	Estudiantes siendo evaluados			
	Esteban Ferarios	**Santiago Gamboa**	**Juan Navarro**	
Esteban Ferarios	—	10	10	
Santiago Gamboa	10	—	10	
Juan Navarro	10	10	—	

VI. TABLA DE ACTIVIDADES POR CADA ESTUDIANTE

Cuadro III
TABLA DE ACTIVIDADES POR CADA ESTUDIANTE

Fecha	**Integrante**	**Actividad**	**Horas**
10-07-2020	Reunión grupal	Investigación de procesos y opciones de interfaz gráfica	2.5
11-07-2020	Reunión grupal	desarrollo de interfaz gráfica primera versión	3
12-07-2020	Reunión grupal	Pruebas de hilos	4
13-07-2020	Reunión grupal	Definición de estructuras para almacenar los objetos	3
14-07-2020	Reunión grupal	Investigación sobre Join de procesos livianos	2.5
15-07-2020	Reunión grupal	pruebas con archivos de configuración	3
16-07-2020	Reunión grupal	Diseño oficial de la interfaz	2
17-07-2020	Reunión grupal	primeras pruebas de	2.5
18-07-2020	Reunión grupal	Creación de Artefactos puentes para administrar recorridos	4
19-07-2020	Reunión grupal	Definición de construcción de mapa mediante configuración	5
20-07-2020	Reunión grupal	Diseño de algoritmos de planificación.	6
21-07-2020	Reunión grupal	Implementación de Round Robin	x
22-07-2020	Reunión grupal	Pruebas de Roun Robin con configuración e implantación de lotería	3
23-07-2020	Reunión grupal	Desarrollo de trayectoria de los aliens	3
24-07-2020	Reunión grupal	Desarrollo de trayectoria del invasor	2
25-07-2020	Reunión grupal	Implementaciones de algoritmos restantes	4
26-07-2020	Reunión grupal	Pruebas a la interfaz	3
27-07-2020	Reunión grupal	Documentación	3

VII. Conclusiones

El ejercicio de este proyecto logró expandir los conocimientos en los temas de procesos pesados y livianos.

Se pudo comprender el impacto, tanto ventajas y desventajas de cada uno de los algoritmos de planificación de los procesos.

VIII. Sugerencias y recomendaciones

- Es valiosos dominar las herramientas de depuración tales como Valgrind y GDB, pues permiten inspeccionar la aplicación de una forma más efectiva que con mecanismos rudimentarios tradicionales con la impresión de logs.
- Realizar los acuerdos mediante la herramienta del modelo de canvas permite tener claro cual tarea está realizando cada miembro. Valgrind es una herramienta interesante para construcción de interfaces gráficas, junto con el potencial del manejo de memoria en C, lo que hace posible la construcción de interfaces para recursos limitados. La metodología de gitflow fue valiosa para realizar los procesos de integración de nuevo código al proyecto.

Anexos

Referencias

[TAN09] ANDREW S. TANENBAUM. *Sistemas operativos modernos.* third. PEARSON EDUCACIÓN, 2009.

[Tea10] Allegro 5 Development Team. *Allegro game programming library*. Ene. de 2010. URL: https://liballeg.org/git.html.

[Amo16] Eric Amodio. *GitLens — Git supercharged*. Ene. de 2016. URL: https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens.

[Mic16] Microsoft. *C/C++ IntelliSense, debugging, and code browsing.* Ene. de 2016. URL: https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools.

[Ben18] Eli Bendersky. *Launching Linux threads and processes with clone*. Ago. de 2018. URL: Launching % 20Linux % 20threads % 20and % 20processes%20with%20clone.

[Gro19] The Open Group. $posix_memalign$. Ene. de 2019. URL: https://pubs.opengroup.org/onlinepubs/9699919799/functions/posix_memalign.html.

[IS19] IEEE y The Open Group Standard. *waitpid(3) - Linux man page*. Ene. de 2019. URL: https://linux.die.net/man/3/waitpid.

[Ker19] Michael Kerrisk. $sched_yield(2)|Linuxmanualpage$. Ene. de 2019. URL: https://man7.org/linux/man-pages/man2/sched_yield.2.html.

[Dev20] Valgrind™ Developers. *Documentation Valgrind*. Ene. de 2020. URL: https://valgrind.org/info/about.html.

[dev20] GDB developers. *GDB: The GNU Project Debugger*. Ene. de 2020. URL: https://www.gnu.org/software/gdb/.

[Gro20] The Open Group. *getpgrp*. Ene. de 2020. URL: https://pubs.opengroup.org/onlinepubs/009695399/functions/getpgrp.html.

[Ker20a] Michael Kerrisk. *getpgrp*. Ene. de 2020. URL: https://man7.org/linux/man-pages/man2/kill.2.html.

[Ker20b] Michael Kerrisk. *getpgrp*. Ene. de 2020. URL: https://man7.org/linux/man-pages/man3/killpg.3.html.

[Ker20c] Michael Kerrisk. *getpid(2) — Linux manual page*. Ene. de 2020. URL: https://man7.org/linux/man-pages/man2/getpid.2.html.

[Ker20d] Michael Kerrisk. $posix_memalign(3)|Linuxmanualpage$. Jun. de 2020. URL: https://man7.org/linux/man-pages/man3/posix_memalign.3.html.

[Ker20e] Michael Kerrisk. *atexit(3) — Linux manual page*. Jun. de 22020. URL: https://www.man7.org/linux/man-pages/man3/atexit.3.htmlLaunching%20Linux % 20threads % 20and % 20processes % 20with%20clone.