Sebastian Butorovic - Gabriel della Maggiora

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2333 — Sistemas Operativos y Redes

Tarea 2: Syscalls de procesos en Pintos

Parte 1: Mensajes de termino

Lo que se hizo para que se imprimieran mensajes de termino con pid y exit code fue añadir un printf en el syscall exit.

Parte 2: Paso de argumentos

Para implementar el paso de argumentos, lo que se hizo fue pasar la linea de comandos desde main en init.c (ya parseada, en forma de argv[]) a process_execute en donde se concatena para formar el string representante de la linea de comandos y el cual se pasa a start_process, despues a load y por último a el metodo setup_stack, en process.c.

En el metodo setup_stack, despues de obtener una página, se llama al metodo copy_arguments, que es el responsable de copiar los argumentos y sus direcciones en el stack. Lo primero que se hace en el metodo copy_arguments es parsear la linea de comandos y copiar cada token en la memoria, para así poder copiarlos en el stack. Despues, copia los parametros al stack en orden inverso siguiendo el formato de la documentación de pintos de Stanford. A continuación, se obtienen las direcciones de memoria en donde fueron recien copiados los parametros en el stack, y se copia esas direcciones de memoria en el stack en orden inverso. Por último, se copia argc (obtenido al momento de parsear la linea de comandos) y un puntero NULL en el stack.

Parte del código del metodo copy_arguments esta basado en un código disponible en internet, en el link: https://github.com/rbell517/Pintos/blob/master/src/userprog/process.c

Parte 3: Llamadas al sistema

Para implementar esta parte lo que se cambió la estructura de threads, agregandoles una lista de procesos hijos.

Halt

Para implementar halt lo único que se hizo fue llamar al metodo shutdown_power_off() en el metodo en syscall.c

Exit

Al llamar al metodo exitThread en el metodo en syscall.c se ejecuta el metodo thread_exit. Si el proceso a terminar es de usuario, se llama a process_exit en donde se le avisa al proceso padre del que se va a terminar que su hijo va a terminar y además se le remueven los hijos al proceso a terminar, quedando huerfanos.

Exec

La forma de implementar exec consiste en llamar a process_execute y asignar como hijo del thread actual el nuevo thread creado en thread_create. Cuando se ejecuta exec el proceso padre espera hasta que el hijo este cargado con la función barrier.

Wait

La forma de implementar exec consiste en que se espera a que el proceso con pid termine, se busca a los hijos del proceso actual y se ve si alguno corresponde al id. Si alguno corresponde, ve si ese hijo ya esta esperando y si ya estaba esperando retorna -2, sino, ve si ya terminó y si terminó retorna el codigo de salida, si no termino, bloquea el proceso para que no haga más acciones y termine. En esta llamada el proceso padre espera a que los hijos terminen mediante la función barrier.