Este documento descreve como rodar o pintos pela 1a vez, uso do DDD e Eclipse para depuração

1. **Configurando e rodando o pintos pela 1a vez**
2. Baixe o código fonte pintos2023.zip. Testado no ubuntu 16 e 22.
3. Instalar o qemu. sudo apt install qemu-system-i386
4. export PATH=$PATH:/home/seu\_usuario/pintos/src/utils **(opcional, veja o item 5).** Além disso, ir no arquivo **‘.bashrc’** (ele aparece quando você pede para mostrar arquivos ocultos em /home) e escrever: PATH=/usr/local/bin:$PATH:/home/seu\_usuario/pintos/src/utils
5. Entrar no diretório src/utils executar make
6. Entrar no diretório src/threads executar make. Ainda em src/threads, ir no arquivo **‘Make.vars’** e trocar ‘SIMULATOR = --boch’ por ‘SIMULATOR = --qemu’

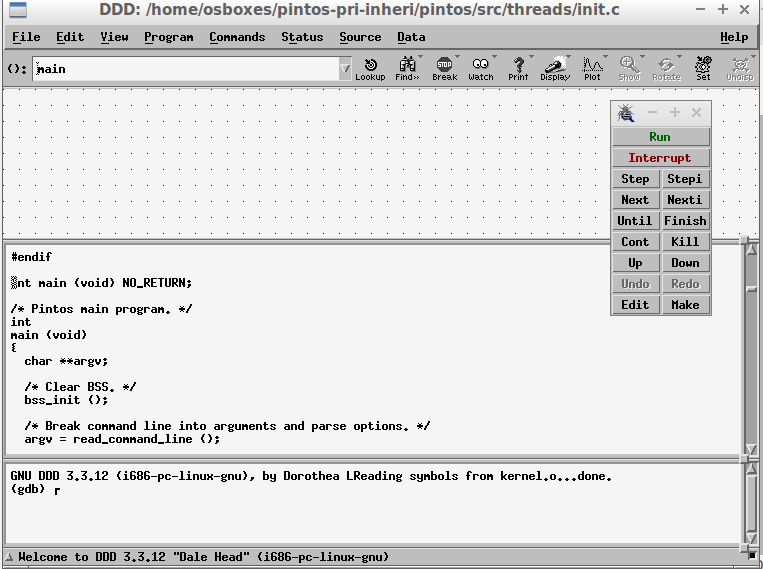
1. Em src/threads, executar pintos --qemu -- run alarm-multiple **(precisa do item 2)** **ou** ../utils/pintos --qemu -- run alarm-multiple (**eu tenho usado este último comando, pois não precisa fazer o export).**Para rodar todos os testes de uma vez, use **make check**  no terminal em threads/build.
2. Documentação: <https://www.scs.stanford.edu/23wi-cs212/pintos/pintos_1.html>

—---------------------------------------------------------------------------------------------------------------

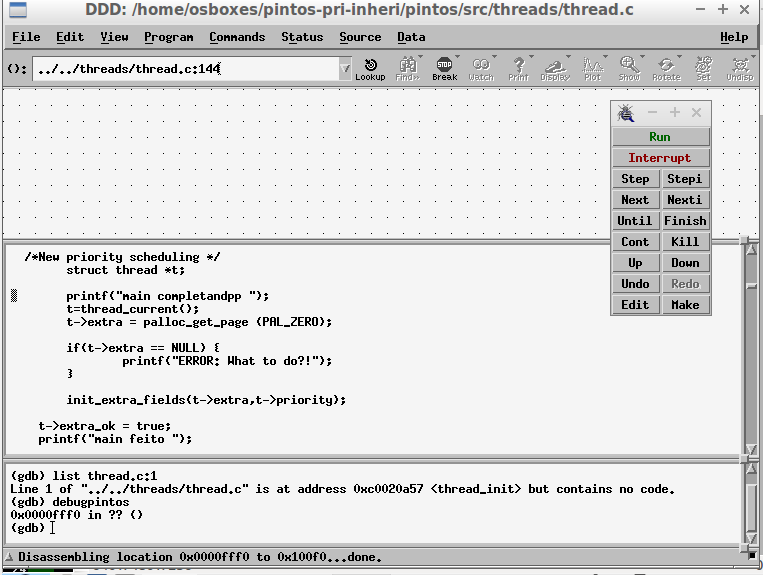
**II. Depuração usando o DDD (Data Display Debugger - GUI do GDB)**

**Obs: as instruções abaixo não consideram atualização de PATH (usando export) para o diretório do pintos. Todavia, pode-se fazer tal alteração. Se precisar, veja o item 1 do tem anterior “Configurando e Rodando…”.**

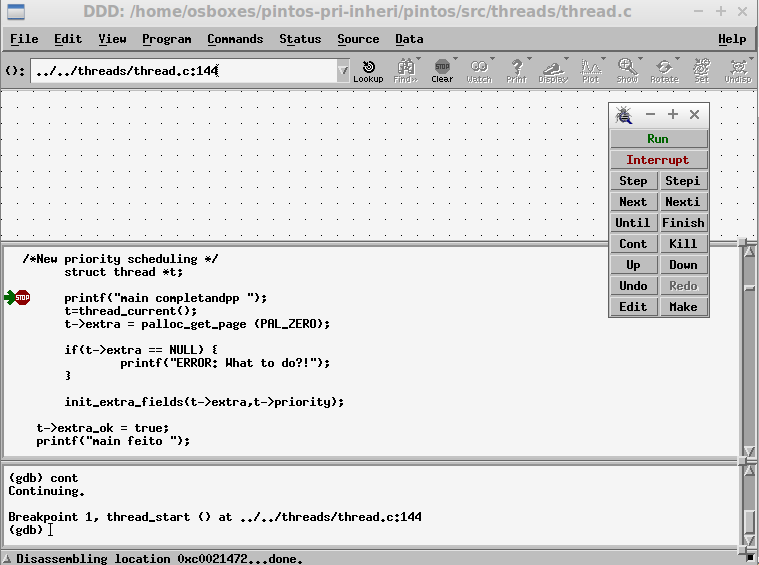
1. Algumas possíveis formas de depuração são explicadas em <https://www.scs.stanford.edu/23wi-cs212/pintos/pintos_10.html#SEC145> (incluindo printf).
2. Para instalar o ddd, sudo apt install ddd
3. Em src/utils, faça cópia de pintos-gdb para ddd-pintos-gdb: cp pintos-gdb ddd-pintos-gdb
4. Abra o arquivo ddd-pintos-gdb usando um editor de texto. Altere a variável GDBMACROS para o local correto onde está o pintos em sua máquina. Ex: GDBMACROS=/home/user-name/pintos/src/misc/gdb-macros
5. Em ddd-pintos-gdb, altere **todas** ocorrências de GDB=... para GDB=ddd
6. Entrar no diretório src/threads e executar make
7. Você vai precisar de 2 terminais separados. Abra um primeiro terminal (shell). Vá para **src/theads**, e execute ../utils/pintos --qemu -v --gdb -– run alarm-multiple. O pintos executará em modo debug e ficará parado
8. Abra um segundo terminal (shell). Vá para **src/threads/build**,eexecute ../../utils/ddd-pintos-gdb kernel.o. A tela abaixo será aberta



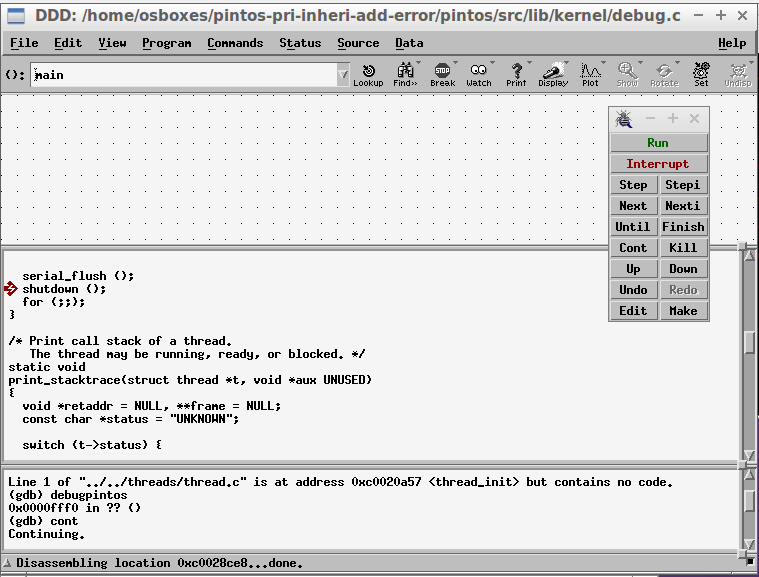
1. Automaticamente, o arquivo src/threads/init.c será aberto. Entretanto, você pode escolher outro arquivo de interesse. Vamos assumir o arquivo threads.c. Dessa forma, vá em File->Open Source->Escolha thread.c
2. No console do ddd (parte de baixo), digite debugpintos. O ddd/gdb fará a conexão com o pintos.



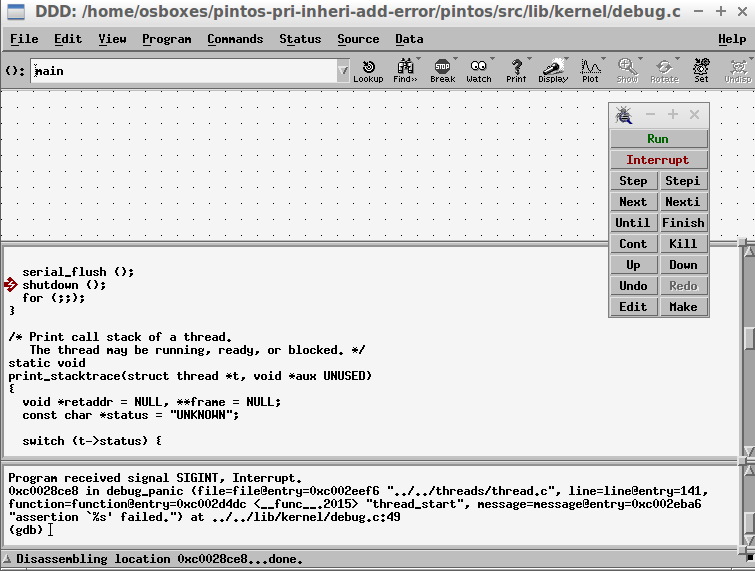
1. Adicione breakpoints. Basta clicar com o botão direito do mouse em cima de um comando desejado. A execução é similar a qualquer debug. Ex: continue (para no próximo breakpoint), step (executa o próximo comando)...



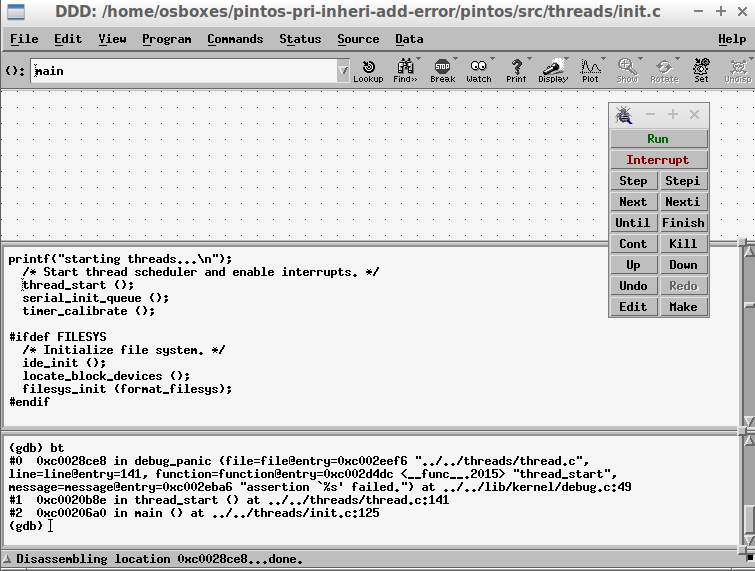
1. Se acontecer algum erro durante a execução do pintos, o ddd poderá ficar travado no continuing ou running



;

Neste caso, basta apertar control-c. 

Digite bt (de backtrace) no console, e ele vai exibir o trace de onde o erro apareceu



Na janela acima, o erro foi na linha 141 de thread.c. Um erro foi adicionado propositalmente para este exemplo

—----------------------------------------------------------------------------------------------------------------------------

**III. Desenvolvendo e depurando usando o eclipse**

**As instruções são baseadas em** [**https://uchicago-cs.github.io/mpcs52030/pintos\_eclipse.html**](https://uchicago-cs.github.io/mpcs52030/pintos_eclipse.html) **e assumem a versão em inglês**

1. Primeiro baixe Eclipse IDE for C/C++ Developers e instale: <https://www.eclipse.org/downloads/packages/> . Eu testei usando a versão 2023-06 (4.28.0)
2. Crie um projeto no Eclipse usando os passos abaixo

* **File** -> **Import** -> **C/C++** -> **Existing Code as Makefile Project**
* Na tela seguinte, coloque os seguintes dados:
  + Project Name: pintos
  + Existing Code Location: O diretório pintos/src deve ser escolhido (é o diretório que contém todos os arquivos, tais como threads, tests, utils)
  + Languages: Deixe “**C” Marcado**. **“C++” Desmarcado**.
  + Toolchain for Indexer Settings: Selecione “Cross GCC”

1. Criando Build Configurations. **Esse item não é necessário, caso você queira continuar (compilando) chamando make na linha de comando**

* Considerando o projeto pintos selecionado no Project Explorer (o gerenciador de arquivos ao lado esquerdo no Eclipse), ir no menu Project -> Properties -> C/C++ Build
* Clique em “Manage Configurations...” and then “New...”. Em “Create New Configuration”, coloque em “Name:” pintos-thread
* Pode deixar “Description” em branco. Ao clicar em ok, volta para Manage Configurations. Selecione pintos-thread e clique em Set Active.
* Ao voltar para janela “C/C++ Build”, selecione a configuração de compilação na lista pull-down “Configuration” para pintos-thread **.** Logo abaixo, em “Build location”, mude o “Build Directory”. Clique em “Workspace…”, e escolha o diretório threads. Você vai ter algo como${workspace\_loc:/pintos/threads}**.** Clique em “Apply and Close
* Para compilar, Project -> Build all. Control-B também funciona. Vai aparecer um erro de compilação no Eclipse, mas não é erro. Tem relação com um warning no make devido ao ld (linker) estar usando um comando obsoleto. Pode ignorar o erro

1. Criando Debug Configurations.

* Depois de fazer o *build* do projeto, procure por threads/build/kernel.o no Project Explorer (o gerenciador de arquivos ao lado esquerdo no Eclipse):
* Clique com o botão direito em cima do arquivo e vá para **Debug As** -> **Debug Configurations...**
* Na janela Debug Configurations, clique 2 vezes em “GDB Hardware Debugging”. Isto vai criar uma nova configuração com alguns campos preenchidos. Altere o Name para pintos-thread
* Em “Build (if required) before launching”, escolha “disable auto build.
* Na aba Debugger, altere “GDB debugger” para:  
  gdb -x /home/local\_onde\_botou/pintos/src/misc/gdb-macros
* Na aba Debugger, opção “Remote Target”. “Use remote target” precisa estar marcado; “Debug server” -> “Generic TCP/IP”; “Protocol” -> “remote”; “Connection” -> “localhost:1234 ”;
* Na aba Start, em “Load Image and Symbols”, **desmarque a opção “Load Image”.** A opção “Load symbols” precisa estar marcada e Use project binary: …/build/kernel.o precisa estar selecionada.

1. Depurando (Debugging)

* Sempre que realizar uma depuração, faça um *build* antes
* Em terminal separado, você vai executar o pintos em modo debug. Vá para **src/theads**, e execute ../utils/pintos --qemu -v --gdb -- run alarm-multiple. O pintos executará em modo debug e ficará parado
* No eclipse, crie algum breakpoint. Por exemplo, abra o arquivo thread.c, em thread\_create, clique 2 vezes no número da linha que tem o comando ASSERT (function != NULL). O breakpoint será criado
* No eclipse, na barra de ferramentas, clique na seta ao lado do ícone do bug para iniciar a depuração pintos-thread . Se não aparecer, vá em “Debug Configurations” para executar a configuração pintos-thread pela primeira vez (nas próximas, ela será a padrão). Ao clicar em Debug, se aparecer “Errors exist in the active configuration…”, ignore e clique em “Proceed”. Clique primeiro no botão resume (Um traço vertical com um play). Em seguida, pode usar os steps ou o próprio resume.