日曜研究室

技術的な観点から日常を綴ります

[xv6 #20] Chapter 1 – The first process – Exercises

テキストの30ページ

本文

1.

swtchにブレークポイントをセットしなさい。
gdbのstepiでswtchのforkretへのretまで一段階ずつ実行し、そしてtrapretを続行するためにgdbのfinishを使い、そして仮想アドレスゼロのinitcodeに到達するまでstepiをしなさい。

2.

実際のOSがどうやってメモリを分けてるかを見なさい。

3.

xv6がスーパーページを使ってなかったら、entrypgdirの正しい定義はどんなだろうか？

4.

execのUnixにおける実装は、伝統的にシェルスクリプトのための特別な操作を含む。
もし、実行したいファイルが#!で始まってたら、その最初の行は、そのファイルを解釈するために実行するプログラムとして扱われる。
例えば、もしexecがmyprog arg1（myprogの最初の行は#!/interpとする）を実行するために呼ばれたら、execは/interpを/interp myprog arg1として実行する。
xv6でこのしきたりをサポートするための機能を実装しなさい。

5.

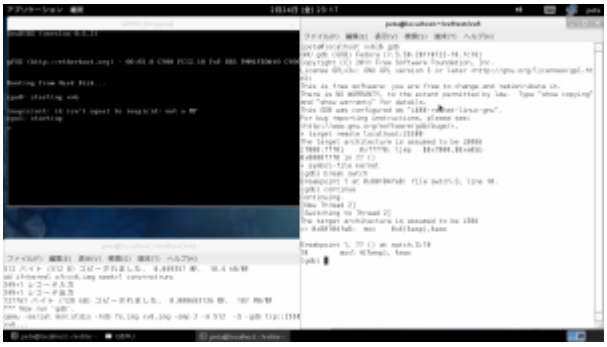
KENBASEは一つのプロセスが利用可能なメモリの量を制限している。
それは4GBフルに積んだマシンではもったいない。
KERNBASEをより上位に持っていく事は、プロセスにより多くのメモリを提供できるだろうか？

作業

1. について

その0でxv6をmakeするために使ったFedoraを使いました。
make qemu-gdb
でgdbによる追跡が可能な状態で実行されます。
警告をエラーと見なす設定になっててうちではそのままコンパイル出来なかったので、とりあえずMakefileの中に書かれている-Werrorパラメータの部分を全部削除したらコンパイル出来ました。
でqemuのウインドウが出ますがgdbからの接続待ちで固まってるので、別のターミナルで
gdb
と実行するとxv6に接続出来ます。

ブレークポイントを設定した段階の画像。



swtchのステップ実行は出来ましたが、initcodeに到達するまでがずいぶん長そうだったので、途中であきらめて
(gdb) break *0
でアドレス0にブレークポイントを置いてcontinueしてしまいました(；・∀・)
確かにアドレス0からの命令を実行してました。

2. について

簡単な各種OSの仮想メモリの対応状況としては、仮想記憶 – Wikipediaにちょっと書いてありますね。
このテキストで今まで読んできたような内容のレベルでの違いを知るためにはソース読んだりしない

と…。

3. について

スーパーページについては1章のCode: entry page tableの節に出てきました。
ここでいうスーパーページとは、4KBごとではなく4MBごとを仮想アドレスに割り当てる起動処理の初期に使われるページの事です。
entrypgdirはmain.cで

```
1 pde_t entrypgdir[NPDENTRIES] = {
2   // Map VA's [0, 4MB) to PA's [0, 4MB)
3   [0] = (0) + PTE_P + PTE_W + PTE_PS,
4   // Map VA's [KERNBASE, KERNBASE+4MB) to PA's [0, 4MB)
5   [KERNBASE>>PDXSHIFT] = (0) + PTE_P + PTE_W + PTE_PS,
6 };
```

として定義されています。
で、これを4KBページ対応にするには、通常と同じようにページディレクトリとページテーブルの2段階構成にする必要があるでしょうね。
がしかし具体的にどう書きなおせばいいかは分かりません(；・∀・)

4. について

実装？Σ(ﾟДﾟ；ｴｰｯ!
一応読むことに重点を置いてるので実装はしませんが、ちょっと考えてみましょう。
xv6のシェルはsh.cで実装されてるので当然そのファイルを変更しなければいけません。
runcmd関数のcase EXEC:の部分の処理に追加すれば良さそうです。
ファイルの1行目を読み込み妥当なシバンならそこからプログラムパスを抜き出し、それをexecの第一引数とし、元のコマンドとパラメータを第二引数として渡せば良さそうです。

5. について

答えはYESでもありNOでもあると思います。
KERNBASEは仮想アドレス空間上の位置に過ぎず、物理メモリ上ではカーネルは最初の方に位置してて、その他諸々を省いた残りをユーザプロセスが分け合うという形になってます。
なので複数のプロセスが使える物理メモリの合計は変わりませんが、一つのプロセスに限れば2GBまで使用可能なのが、いくらか増やせるかもしれません。

感想

Exercisesはほとんど答えられないんじゃないかと思ってました。
妥当な内容かどうかは別としてこれだけ反応することが出来るようになった事がとても嬉しいです。

このテキストを読み始める前の自分だと手も足も出なかったかもしれません。

あとOS界隈の方には常識なのかもしれませんが、xv6の起動プロセスをgdbでデバッグ出来る事に感動してしまいました！

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/2/25 土曜日 [http://peta.okechan.net/blog/archives/1339] |
