日曜研究室

技術的な観点から日常を綴ります

[xv6 #11] Chapter 1 – The first process – Address space details

テキストの20〜21ページ

本文

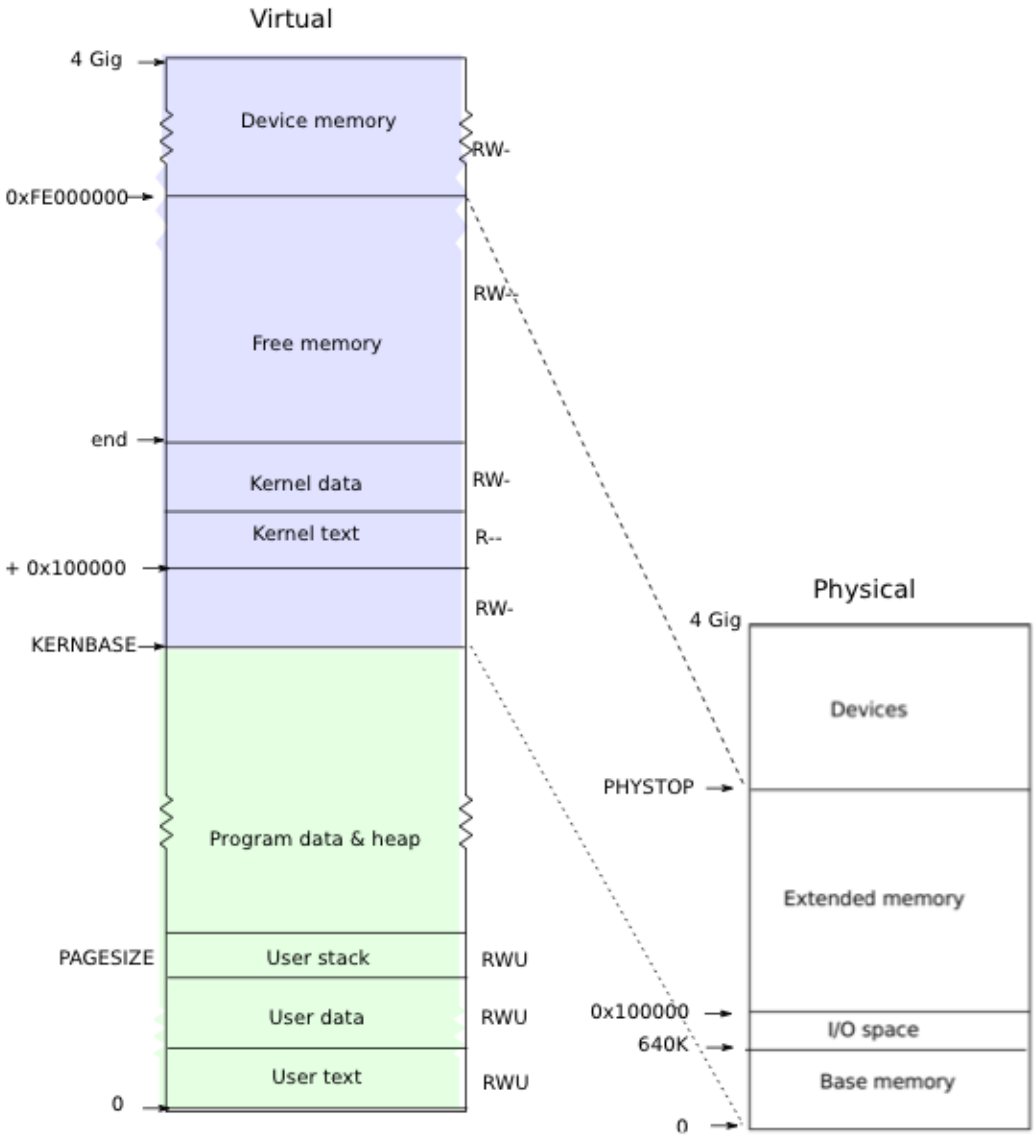
entryルーチンによって作られたページテーブルは、カーネルのCコードを実行しはじめるのに十分である。
しかしながら、main関数はただちにkvmalloc（vm.c）を呼んで新しいページテーブルへと変更する。
なぜならカーネルは、プロセスのアドレス空間を表現するため、ページテーブルについてもっと複雑なプランを持ってるからである。
（これだけ見ても詳細は分かりませんが一応vm.cのkvmalloc関数の部分だけ載せときます。）

```
1 // Allocate one page table for the machine for the kernel address
2 // space for scheduler processes.
3 void
4 kvmalloc(void)
5 {
6   kpgdir = setupkvm();
7   switchkvm();
8 }
```

どのプロセスも分離されたページテーブルを持つ。
xv6が実行中のプロセスを切り替えたときに、ページテーブルも切り替えるようxv6からページテーブルハードウェアに伝えられる。
図1-2で示されるように、プロセスのユーザメモリは仮想アドレス0から始まり、KERNBASEまで拡張できる。
つまりひとつのプロセスで最大2GBまでのメモリを参照できる。
プロセスからxv6へメモリ追加の要求があったとき、xv6はまず、空いている物理ページを探し、その空いている新しい物理ページを指し示すPTE群をそのプロセスのページテーブルに追加する。
（要求されたメモリのサイズによって、PTEが一つで済むとは限らない）
xv6は、そのPTE群にPTE_U、PTE_W、PTE_P（ユーザ用、書き込み可能、有効）フラグをセットする。
ほとんどのプロセスは、ユーザアドレス空間全体は使わない。

xv6は、使ってないPTE群に対してはPTE_Pフラグをクリアしたままにする。
他のプロセスのページテーブルは、ほかのユーザアドレスを物理メモリ上の他のページに変換する。
それは、全てのプロセスが個別のユーザメモリを持てるようにするためである。

図1-2 仮想アドレス空間と物理アドレス空間のレイアウト



xv6は、どのプロセスのページテーブルでもカーネルが動作するために必要とされる全ての対応付けを含む。
そのような対応付けは全て、KERNBASEより上に現れる。
それは、仮想アドレスKERNBASE:KERNBASE+PHYSTOPを物理アドレス0:PHYSTOPに対応付ける。
こうなってる理由の一つは、カーネルがそれ自身の命令とデータを使えるようにするためである。
他の理由は、しばしば（例えば、ページテーブル用のページを作成するときに）、カーネルは与えられた物理メモリ上のページに対して書き込める必要があるからである。
全てのプロセスが、こういう仮想アドレスに現れる予測可能な物理ページを持つことは、このような

便利さを創りだす。
この決まりごとの欠点は、xv6が2GB以上の物理メモリをプロセスに対して使えるようにできない点である。
メモリマップドI/Oを使ういくつかの機器は、0xFE000000から始まる物理アドレスを使う。
なのでxv6のページテーブルはそういう機器のための直接の対応付け（direct mapping）を持つ。
xv6はPTE_UフラグをKERNBASEより上を指すPTE群にはセットしない。
それは、その領域をカーネルからのみ使えるようにするためである。

プロセスごとに、ユーザメモリとカーネル全体の両方を対応付けるページテーブルを持つ事は、システムコールや割り込み中にユーザコードからカーネルコードへ切り替えるときに便利である。
そのような切替は、ページテーブルの切替は必要としない。
大部分はカーネルは自身のページテーブルを持たない。
それは大抵、どこかのプロセスのページテーブルを借りている。

おさらい。
xv6は、どのプロセスも自分自身のメモリだけを使えるようにする事と、プロセスに自身のメモリを連続した仮想アドレスとして見せるようにする事を確実にする。
まず、プロセス自身のメモリを参照する仮想アドレスのPTE群に対してのみPTE_Uフラグを設定する事によって、xv6はそれを実装している。
次に、連続した仮想アドレスをプロセスに割り当てられたどんな物理ページにでも変換するために、ページテーブルの能力を使う事によって、それを実装している。

感想

アドレス空間についてです。
図1-2が全てなんですが、何故そうなのかなどが色々説明されてます。

カーネルは、通常のプロセスと同じように考えた場合に必要となるページテーブルの大部分を持たないらしいですが、全てのプロセスのKERNBASE以降に固定的にアクセスすることによって、自身のページテーブルを持たないまま動作することが出来ると。

2GBってところは多分KERNBASEが0×80000000だからですね。

翻訳って難しい。
いやそもそも翻訳というよりは、自分で読んでるときのメモをそのまま公開してる感じに近いんですが、内容は分かっても日本語でどう現していいのかしっくり来ない事が多いです。
思いっきり言い換えれば自然な日本語に出来るんでしょうけど、理解も中途半端なので下手に言い換えるよりは原文の雰囲気を残すように直訳風味にしたほうがいいのかななんて思いながらやってます。

カテゴリー: 技術 | タグ: xv6 | 投稿日: 2012/2/16 木曜日 [http://peta.okechan.net/blog/archives/1268] |