東京科学大学情報理工学院 総合型選抜　　活動実績報告書

氏名（よみ):　**大岡山 花子（おおおかやま はなこ）**

高等学校:　　**〇〇県立△△高等学校**

(2025年 3 月　☐卒業・☒卒業予定)

活動実績概要（150 字程度):

オペレーティングシステム（以下，OS）の仕組みの勉強のために，小さな OS udos を作成した．その際，IA-32 でページング機能オン後に，カーネルが物理アドレスにアクセスするための方法に疑問を持ち調べた．結論として同一マップ領域と自己マップ領域があることを知り，udos でもその方法を用いた．

活動実績の実施状況:

☒志願者が単独で行った

☐教師などからの指導を受けながら志願者が単独で行った

☐共同で行った

☐その他

報告書本体ページ数(表紙を含まない): 3 ページ

注意:

- 報告書本体を 4 ページ以内で作成し、この表紙と一緒に提出すること。
- 報告書本体の形式は自由とするが、報告書の題目を14ポイント以上の文字で必ず記載し、それ以外の文字の大きさは 10 ポイント以上にすること。また内容として活動実績の背景、具体的な内容、活動実績の実施状況の説明、参考にした資料の一覧などを必ず含むこと。
- 報告書本体に、活動実績を志願者が単独で行ったか否か、共同で行った場合は自身の役割、指導を受けた場合はどの部分に対する指導か等の説明を書くこと。
- 報告書本体には氏名、学校名はどうしても必要な場合を除いて書かないこと。

小さなオペレーティングシステム udos の作成

概 要

オペレーティングシステム（以下，OS）の仕組みの勉強のために，小さな OS udos を作成した．ここでは得られた様々な経験の中から，特に次の経験を述べる：

- **経験**：IA-32 でページング機能オン後に，カーネルがどうやって物理アドレスにアクセスするのかという謎が解けた．

1 はじめに

1.1 動機：OS の仕組みは謎だけど興味深い

OS はプロセス，メモリ，ファイルシステム，ネットワークなどコンピュータ上の様々な資源を管理する．例えば各プロセスは**仮想メモリ**の機能によりプロセスごとに独立した**仮想メモリ空間**を持つ．このため仮にプロセスが暴走しても，一般的に他のプロセスのメモリを破壊することはない．

この仮想メモリの仕組みは原理的には簡単である．物理メモリ上の**ページテーブル**という特別な領域に，**仮想アドレス**と**物理アドレス**の対応表を置いておく．するとメモリアクセス時に CPU のハードウェア機能 (MMU, memory management unit) がページテーブルを使って，仮想アドレスを物理アドレスに変換して，物理メモリに自動的にアクセスしてくれるのだ．

しかし，IA-32 ではページング機能をオンにすると，カーネルでさえ，直接，物理アドレスにはアクセスできなくなる．カーネルはどうやって物理メモリ上にあるページテーブルやハードウェアの固定物理アドレスにアクセスするのか？ページテーブル自身もページングの対象にするのか？考え始めると，これ以外にも，OS の仕組みに興味が湧いてきたため，私は小さな OS を作成して，OS の謎を解くことにした．

1.2 作成方針：リアル，コンパクト，明確な根拠，標準ツールの利用

作成にあたり，以下の作成方針を立てた．

- **リアル**：PC/AT 互換機（いわゆるパソコン）で実際に起動・動作すること，もちろん，コーディング中は Bochs や VMware などの仮想マシンも使用するが，最終的には実機での動作を目標とした．
- **コンパクト**：有名な教育用 OS である MINIX は主要部分（src/{kernel，mm}）だけでも 3 万行もあり非常に大きい．そこで今回はコンパクトな実装を目指すことにした．具体的には，実行性能，移植性，多くのデバイスのサポートはせず，機能の絞り込み，単純な実装方法の採用，エラー処理の省略などで，実装を小さくする．
- **明確な根拠**：例えば，最初の Linux [2] のソースコードを読んで，内容を理解せず，コードだけ模倣して動作できても，OS の謎は解けない．CPU，ハードウェア，PC/AT 互換機の規格（例：[3] [4] [5]）をなるべく参照して「なぜこう書くとうまく動作するのか」の根拠が明確になるように心がける．
- **標準ツールの利用**：例えば，バイナリファイル中の各セクションのデータ編集用に，独自ツールを開発することもまた勉強になりうる．しかし，今回は範囲外と割り切り，バイナリの編集には GNU Binutils や GNU BFD という標準的なツールを使用する．

2 作成した OS udos の概要

概ね，作成方針通りに udos を作成できた．udos のソースコードは Web 上で公開している [6]．udos の主な機能を以下に列挙する．

- PC/AT 互換機上（BIOS）で動作（USB メモリに OS イメージをコピーして起動）．
- プリエンプティブ・マルチタスク（カーネル自体はノンプリエンプティブ・カーネル）
- ページングによる仮想メモリ
- FAT12 ファイルシステム
- fork や exec など，POSIX API サブセット

図 1 に udos の実行画面を示す．ここでは仮想マシン Bochs 上での実行画面だが，もちろん実機でも動作する．図 2 は udos の構成図である．図中で上にある要素は下の要素にソフトウェア的に依存している．表 1 に udos

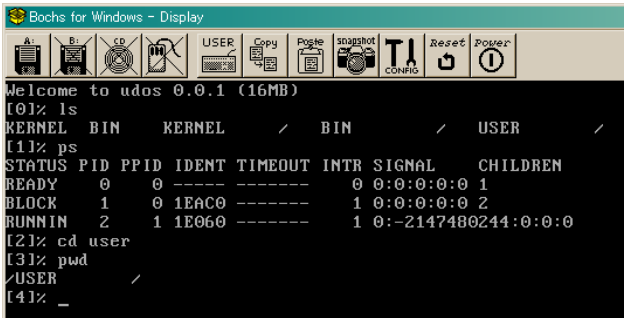

図 1: Bochs 上での udos の動作画面

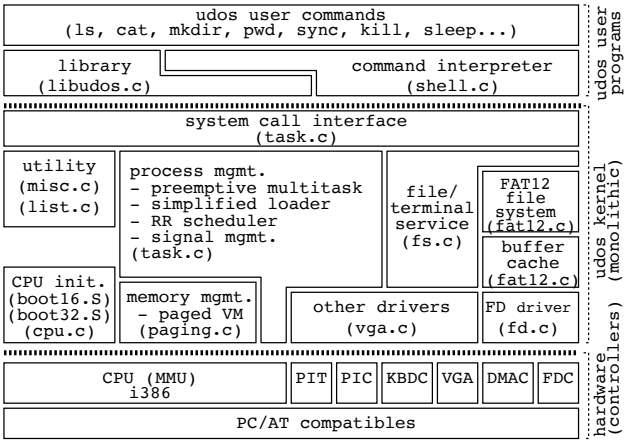

図 2: udos の構成図

の行数を示す．合計で約 5500 行であり，MINIX の 3 万行 [1]，最初の Linux [2] の 1 万行に比べると，非常にコンパクトに実装できた．

3 ページング機能下での物理メモリへのアクセス方法

答を先に書くと以下の通り．

- **答**:物理アドレスにアクセスするには，アドレス空間の一部を固定的に，物理アドレスと仮想アドレスを**同じアドレス**にマップする (identity mapped page table)．また，ページテーブルにアクセスするために，ページテーブルを仮想アドレス空間にマップする．**ページテーブル自身をマップするページテーブル** (self-mapped page table) を使えば，これを簡単に実現できる．

表 1: udos の構成ファイルの行数

ファイル名	行数	内容
boot16.S	321	ブート処理（保護モードに入る前）
boot32.S	135	ブート処理（保護モードに入った後）
main.c	58	udos メイン関数
cpu.c	169	CPU 初期化（GDT や IDT の設定）
fat12.c	1233	FAT12 ファイルシステム・ツール
fd.c	554	フロッピードライバ，バッファキャッシュ
fs.c	171	file 構造体，システムコール（ファイル）
list.c	106	簡易汎用リスト処理関数
misc.c	146	その他（printk，panic，memcpy など）
paging.c	561	ページングによる仮想メモリ
task.c	740	プロセス管理，システムコール（プロセス）
vga.c	604	VGA などの入出力デバイス管理
exe2bin.c	105	ロード簡易化ツール

3.1 ページング概要

ページングとは固定長のメモリブロック（例：4KB）で仮想メモリを管理する方法である．このメモリブロックを**ページ**と呼ぶ．各プロセスごとに異なるアドレス空間を与えるため，通常はプロセスごとにページテーブルを切り替える．

ページングではページテーブルが大きくなりがちなので，多段（マルチレベル）のページテーブルを使用する．IA-32 では 2 段（2 レベル）のページテーブルを使用する（図 3）．

32 ビットアドレス，ページサイズが 4KB の場合，上位 10 ビットが 1 段目のページテーブル（ページディレクトリと呼ぶ）のインデックス，次の 10 ビットが 2 段目のページテーブルのインデックスであり，得られたページの先頭物理アドレスに，下位 12 ビットをオフセットとして加算して，最終的な物理アドレスを得る．

3.2 identity mapped page table

udos では，仮想メモリ空間の先頭 4MB の領域（**同一マップ領域**と呼ぶ）の仮想アドレスと物理アドレスを一致させた．これには次の利点がある．

- 同一マップ領域では，仮想アドレスをそのまま用いて，物理メモリにアクセスできる．VRAM へのアクセスなど，ハードウェアの固定アドレスへのアクセスに便利．
- IA-32 では同一マップ領域が必須な場合があり，その制約に対応できる．例えば，mov cr0 命令と直後の jmp 命令は同一マップ領域中でなくてはいけないという制約がある [3]．
- 同一マップ領域にカーネルコードを置けば，簡単に各プロセスでカーネルコードを共有できる（もちろんユーザ空間からはアクセス不可に設定する）．

3.3 self-mapped page table

図 4 に示す通り，ページテーブルがページテーブルを指すように設定すると，その領域（図 4 では最後の 4MB の領域，**自己マップ領域**と呼ぶ）はページテーブルをマップした領域になる．つまり，カーネルは仮想メモリを使ってページテーブルにアクセスできる．

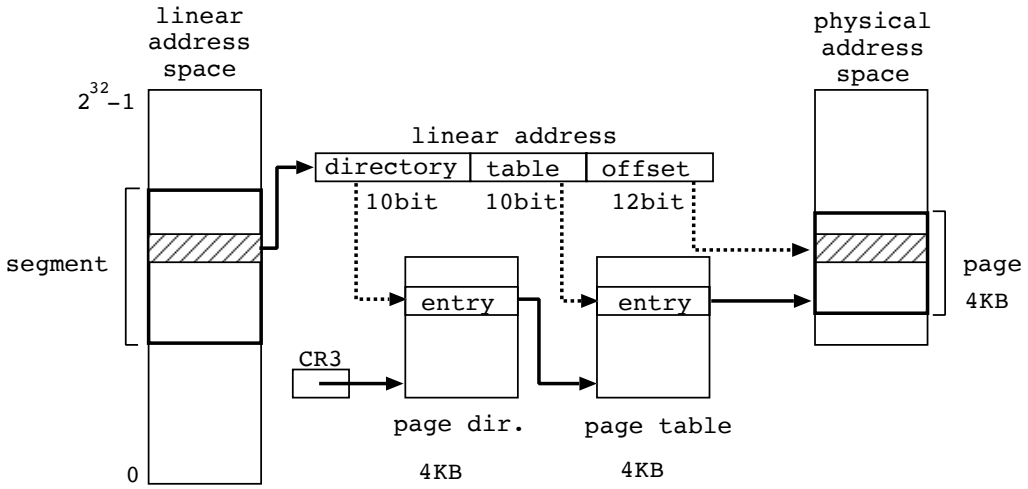

図 3: IA-32 のページングの概要

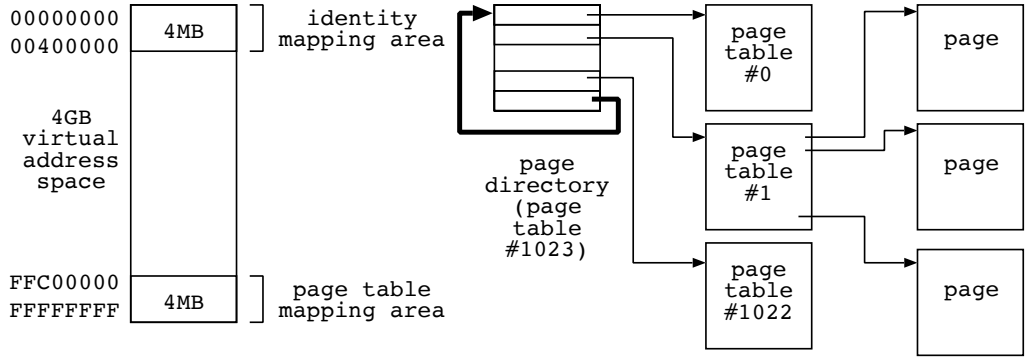

図 4: ページテーブルの自己マップ

4 おわりに

OS の謎を解くために，小さな OS udos を実装した．ここではページング機能下で，カーネルが物理メモリにアクセスする方法として，同一マップ領域と自己マップ領域を説明した．これらのテクニックは OS の設計・実装本（例：[8]）を参考にした．謎の答えが，びっくりするような複雑な機能や工夫ではなく，想像よりも遥かに単純だったことが興味深い．

なお，2017 年現在，東京工業大学 情報工学系の授業では xv6 [7] という小さな教育用 OS（約 6000 行）を使った OS 講義が渡部卓雄教授により行われており，OS の謎を実践的に学ぶことができる．

参考文献

[1] MINIX information sheet, http://www.cs.vu.nl/~ast/minix.html

[2] L.B. Torvalds, Linux 0.01 kernel sources, 1991, http://www.kernel.org/pub/linux/kernel/Historic/linux-0.01.tar.gz

[3] Intel, インテル・アーキテクチャ・ソフトウェア・ディベロッパーズ・マニュアル, 1999.

[4] Intel, 8259A programmable interrupt controller, 1988.

[5] PC Open Architecture Developer's Group，OADG テクニカル・リファレンス DOS/V BIOS インターフェース 技術解説編 第 2 版，1994.

[6] udos Homepage, http://www.sde.cs.titech.ac.jp/~gondow/udos/index.html

[7] xv6 Homepage, https://pdos.csail.mit.edu/6.828/2012/xv6.html

[8] D. Eranian, S. Mosberger: IA-64 Linux Kernel: Design and Implementation, ISBN-10: 0130610143, Prentice Hall, 2002.