RPG のような世界で動く OS の本

［著］大神祐真

技術書典 18 新刊
2025 年 5 月 31 日　ver 1.0

■免責

本書は情報の提供のみを目的としています。
本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

■商標

本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。
また本書では、™、®、© などのマークは省略しています。

はじめに

本書を手に取っていただき、ありがとうございます。本書では、RPG のような世界で動く OS「RPGOS」について解説します。

背景

当サークル（へにゃぺんて）では「独自な OS を作る」ということを主な活動の軸としておりまして、RPGOS もその取り組みの一環です。きっかけは前述の「RPG のような世界の上で動く OS」という思いつきですが、そのような思いつきに至った経緯として、「独自な OS」についての考え方があります。何によってその OS を「独自である」とするかですが、考え方の一つとして、「独自なハードで動く」ということが挙げられます。例えば「他に無い要素を備えた独自なハードで動作する OS」などはまさに「独自な OS」かと思います。

話は変わって、複雑なものの内部の仕組みの説明を省くときなどに「中では小人さんががんばっている」という表現をすることがあります。それが嘘ではなく本当に「中で小人さんががんばっている OS」として、「見た目は単なる黒背景に白文字のシェルだが、その裏では RPG のような世界が広がっていて、その中の NPC[*1]ががんばることで OS が成り立っている」というのはどうだろうという考えがありました。

このような思いつきや考えから、RPGOS を製作しています。

本書の更新情報などについて

本書を含め、当サークルの同人誌・同人作品の情報は下記の筆者ウェブページにまとめています。

- `http://yuma.ohgami.jp/`

本書の内容について訂正や更新があった場合もこちらのページに記載します。何かおかしな点があった場合などは、まずこちらのページをご覧ください。

[*1] 「Non Player Character」の略で、プレイヤーが直接操作しないキャラのことです。

目次

第 1 章
RPGOS の使い方と概要

この章では、まずブラウザ上で動作する RPGOS の使い方として、アクセス方法や操作方法を紹介します。そして、RPGOS がどのような世界観の上に成り立っているのか、その概要を紹介します。

1.1 PLiCy 内の RPGOS のページへアクセス

RPGOS は、PC のブラウザ[*1]で動作するものを、PLiCy（プリシー）というサイト内の下記の場所で公開しています。

- PLiCy 内の RPGOS の公開ページ
  - `https://plicy.net/GamePlay/202689`

まず、上記の URL にアクセスしてください。あるいは、PLiCy のトップページで「RPGOS」というキーワードで検索しても、上記のページへたどり着けます。

[*1] 現状、スマートフォンやタブレットには対応していません。

すると図 1.1 の画面が開きます。（説明文などは変わっているかもしれません。）

▲図 1.1: RPGOS の公開ページ

1.2　RPGOS を起動する

開いた画面内の「大画面でゲームを遊ぶ」をクリックすると図 1.2 の画面が開きます。

▲図 1.2: 実行画面

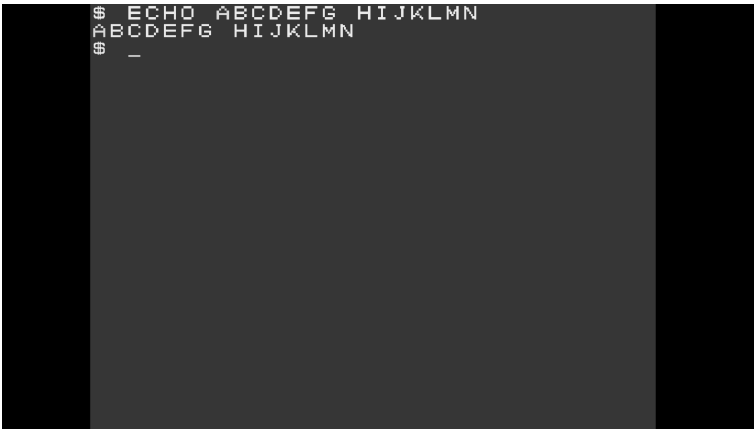
これが RPGOS の実行画面です。なお、RPGOS の画面サイズは 1366x768 px です。ブラウザのウィンドウがそれよりも大きい場合、その他の領域は単に白く表示されているかと思います。

1.3　シェル上でコマンド操作をしてみる

この実行画面は一応シェルになっていまして、キーボードで文字を入力すると、ちょっとラグがありつつも、入力した文字が表示されます。なお、対応しているキーはアルファベット（大文字のみ）、スペース、ドット（.）、ドル（$）、Enter、バックスペースのみです。アルファベットはシフトキー押下の有無に関わらず大文字として扱われます。

♣ ECHO コマンド

試しに `ECHO` コマンドを実行した様子は図 1.3 の通りです。



▲図 1.3: ECHO コマンドを実行した様子

♣ 使用可能なコマンド

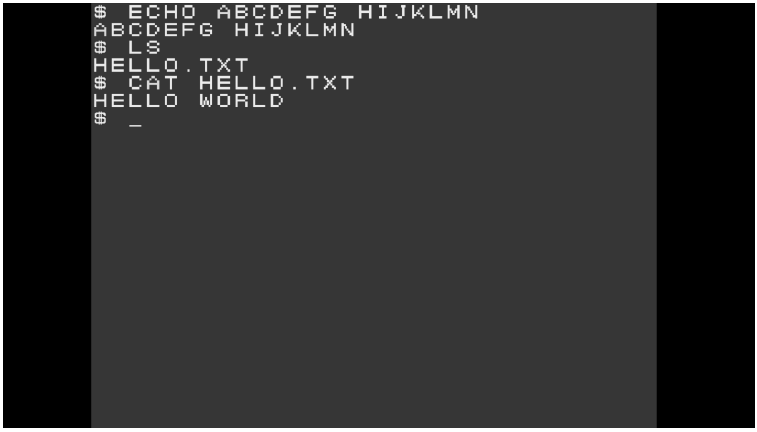
まだ RPGOS で使えるコマンドはわずかです。全てリストアップしても表 1.1 の通りです。

▼表 1.1: 使用可能なコマンド

書式	機能説明
ECHO <文字列>	<文字列>を出力する。
LS	カレントディレクトリにある ファイル/ディレクトリを出力する。 引数は指定しても無視される。
CAT <ファイル名>	<ファイル名>の内容を出力する。 引数無しでの実行はエラー。
SHOWINSIDE	裏側の RPG 世界を表示する。
HIDEINSIDE	裏側の RPG 世界を隠す。（デフォルト）
NPCSLOW	NPC がゆっくり動くモードにする。
NPCFAST	NPC が速く動くモードにする。（デフォルト）

♣ LS と CAT

他に、`LS` と `CAT` も試してみましょう（図 1.4）。



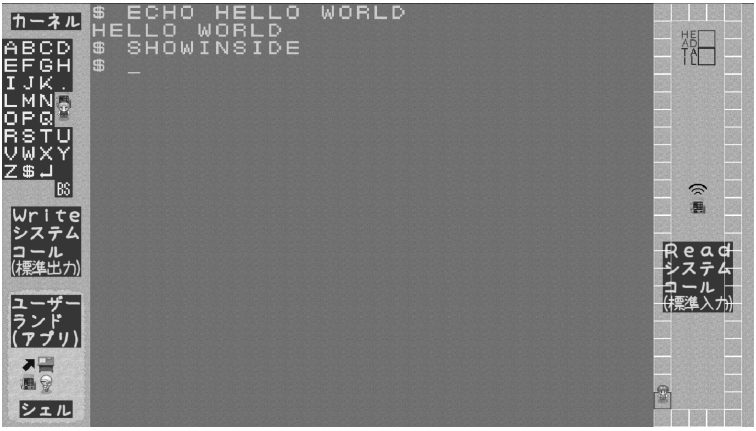
▲図 1.4: LS と CAT を実行した様子

現状、カレントディレクトリには「HELLO.TXT」というファイルが存在するのみで、その内容が「HELLO WORLD」である事がわかります。

♣ 独自コマンド

表 1.1 の `SHOWINSIDE` 以降のコマンドは RPGOS 独自のコマンドです。

試しに `SHOWINSIDE` を実行すると、シェルが動いていた裏側の世界を見ることができます（図 1.5）。裏側は RPG のような世界になっていて、これがまさに「RPG のような世界で動く OS」ということです。（詳しくは次章以降で解説します。）



▲図 1.5: SHOWINSIDE コマンドを実行した様子

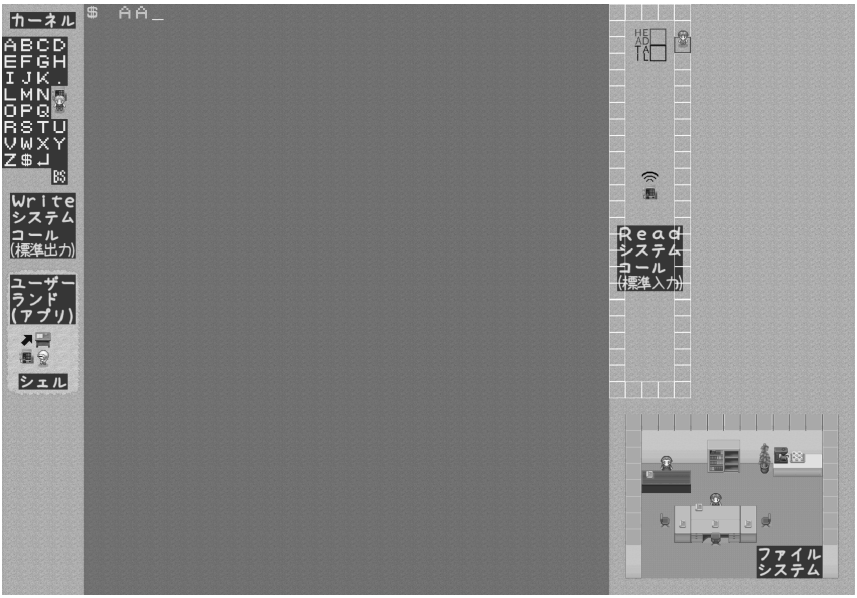
`HIDEINSIDE` を実行すると、元の表示状態である RPG 世界が隠された状態に戻ります。

`NPCSLOW` / `NPCFAST` は裏の RPG 世界を表示した状態で使う用途のコマンドです。キー入力を行うと RPG 世界の上で NPC が動き回るのが見えますが、デフォルトでは動く様子を見るには速いです。 `NPCSLOW` コマンドを実行すると NPC の動きがゆっくりになります。 `NPCFAST` コマンドを実行すると元の速く動くモードに戻ります。

1.4　その他のユーザー操作

ユーザーが他にできる操作として「カメラ移動」があります。初期状態で見えている範囲がこの世界の全てではなく、キーボードの十字キーで表示範囲を縦横に移動できます。カメラを移動すると、初期状態では見えていない OS の構成要素などを見ることができます。

なお、現在の RPGOS のマップ全体は図 1.6 の通りです。



▲図 1.6: マップ全体

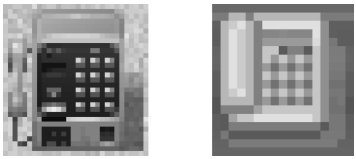
1.5　ユーザーランドとカーネルとシステムコール

裏側の RPG 世界は、ユーザーランドとカーネルに領域が分かれています。分かれていると言っても、現状は地面の色が違うだけですが、図 1.7 がユーザーランドの領域です。

▲図 1.7: ユーザーランド領域

そして、ユーザーランド領域のアプリケーションは、システムコールを通じてカーネルが管理しているリソースへアクセスします。例えば、画面中央のスクリーンへ文字を出す際は、標準出力へのシステムコールを行います。

RPGOS では、この世界の上でシステムコールを「電話」で表現しています（図 1.8)。ユーザーランド領域にも、対向のシステムコール側にもそれぞれ電話があります。ユーザーランドのアプリケーションが何らかのシステムコールを行う際は、そのシステムコールへ電話をかけます。



▲図 1.8: RPGOS に存在する電話の例

電話をかけるのも、電話を受けて対応する処理を行うのも、すべて NPC の役割です。詳しくは次章から機能ごとに解説します。

第 2 章
Write システムコール（標準出力）

この章では、「Write システムコール（標準出力）」について解説します。

2.1 どのようなシステムコールか？

指定された文字あるいは文字列を、標準出力（スクリーン）へ出力するシステムコールです。

RPGOS では、まだ Write システムコールはファイルに対応していないため、出力先は標準出力のみです。

2.2 構成要素

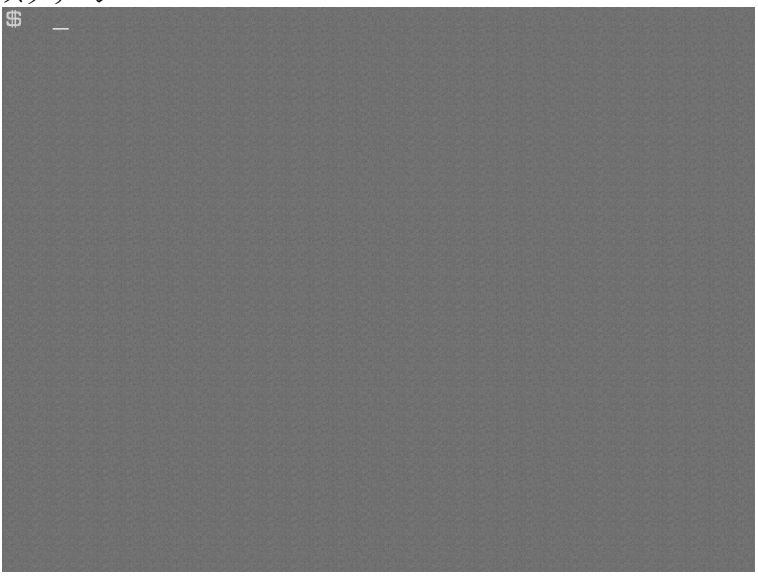
Write システムコールは、以下の要素で実現されています。

- NPC

- 電話

- 文字パネル

- スクリーン



- テキストカーソル

なお、電話の上に状態を示すアイコンが出ることがあります。各アイコンの意味

は以下の通りです。（残念なことに Write システムコール領域の電話の直ぐ上には文字パネルがありアイコンがちょっと見づらいため、このスクショは次章で紹介する Read システムコールのものです。ただ、出るアイコンと意味は同じです。）

電話の状態を示すアイコン

着信中	通話中（電話を受けた側）
	

2.3 動作の流れ

♣ 1. システムコールの電話を待つ

Write システムコール担当の NPC は、初期位置として電話の前にいます。そして、システムコールの電話がかかってくるのを待ちます。

♣ 2. 電話に出て、文字/文字列を聞き、電話を切る

電話がかかってきたら出て、電話相手から標準出力（画面中央のスクリーン）へ出力する文字あるいは文字列を聞きます。聞き終えたら電話を切ります。

♣ 3. 聞いた文字の文字パネルを取得

相手から聞いた文字（文字列の場合、まだ出力していない先頭文字）の文字パネルが置かれている場所へ向かいます。そして、該当の文字パネルを 1 枚取得します。見た通り、文字パネルは巨大で NPC 一人分くらいの大きさがあるので 1 枚ずつしか持てません。

♣ 4. スクリーン上のテキストカーソルが示す座標へ文字パネルを配置

スクリーン上のテキストカーソルが示す座標まで移動し、取得した文字パネルをそこへ配置します。文字パネルを配置するとテキストカーソルが自動的に 1 文字分移動します。

なお、イメージとしては、このスクリーンは NPC の頭上にあって、1 文字ずつ文字パネルがはめ込まれています。（文字が無い所も空白文字の文字パネルがはめ込まれています。）文字を出力する際は、既にはめ込まれているパネルを外し、取得した文字パネルをはめ込む、ということを行っている想定です。ちなみに、その際にテキストカーソルは、スクリーン上の文字パネルをはめ込むレールを、アメンボのような形状でまたいで滑って動いている想定です。

♣ 5. 文字列の場合、文字パネル取得へ戻る（3 へ戻る）

文字列の出力でかつまだ出力していない文字が残っている場合、次の文字を出力するため、3 に戻ります。

1 文字だけの出力であるか、文字列であるが全ての文字を出力し終えている場合、次にすすみます。

♣ 6. 電話の正面に戻り、再び電話を待つ（1 へ戻る）

相手から依頼された文字あるいは文字列の出力を終えたので、電話の正面の位置に戻り、またシステムコール依頼が来るのを待ちます。すなわち、1 へ戻ります。

【コラム】RPGOS の Write システムコールは同期型

電話を待っているフェーズ以外では、電話がかかってきても出ません。そのため、システムコールを依頼する側が電話をかけた時、まだ Write システムコール担当が作業中である場合、システムコールを呼び出した側は相手が電話に出るのを待つことになります。そのため、RPGOS の Write システムコールは同期型（ブロッキング）という感じです。

2.4　現状の仕様や課題、小ネタなど

♣ スクリーンは縦方向にどこまでも長い

スクリーンはマップ全体の高さ分の長さで作っています。今後、OS の要素を追加していくに連れ、マップはどんどん広くなっていきますが、その際にスクリーンはマップの高さに応じてどんどん長くなっていく想定です。そして、イメージとしては、この惑星の 1 周分の長さのスクリーンがあるという想定でいます。

そのため、テキストカーソルが表示されている領域の最終行まで達した際の自動スクロールとしては、単にカメラが 1 文字分下へ移動するというだけです。上方向にスクロールしたい場合は、単にカメラ移動として上キーで移動すれば良いです。

ただ、マップを広げ、マップ内のオブジェクトを増やすに連れ、この RPGOS というブラウザ上で動くソフトを実行する PC のメモリ使用量もさることながら処理も重くなっていくので、現実的にどう実装していくかは要検討です。

♣ 1 文字の出力にかかる時間

スクリーン上の、Write システムコールの電話や文字パネルが置かれている領域から遠い場所へ文字を出力する場合、1 文字の出力に時間がかかります。テキストカーソルが下方向へ進むに連れ、それはどんどん悪化していくことになります。

まだ検討中ではありますが、電話や文字パネルの領域を適宜移動させるとか、そのような領域は点々とあり、テキストカーソルがある地点より下になると次の領域の担当へ引き継ぐ、とかぼんやりとアイディアはあります。

【コラム】エンジニアでなくても「わかる」OS

この章で見たように RPGOS では、OS の機能を RPG のような世界の上で、基本的に「人力」の形で表現しています。「裏側は実は人力だった」というのは漫画やアニメではギャグのような形で用いられることがありますが、それを OS でやってみようという思いつきです。

きっかけは思いつきでしたが、「人力で各種の機能が成り立っている」という世界は、エンジニアではない方にもわかってもらえる OS にできるのではと考えています。（ただ、ごちゃごちゃしてしまうので、見せ方に工夫は必要ですが。）

第 3 章
Read システムコール（標準入力）

この章では、「Read システムコール（標準入力）」について解説します。

3.1 どのようなシステムコールか？

ユーザーのキー入力（標準入力）を 1 文字返すシステムコールです。

Read システムコールはファイルにも対応していますが、その場合の動作は後の「ファイルディスクリプタ」の章で解説します。

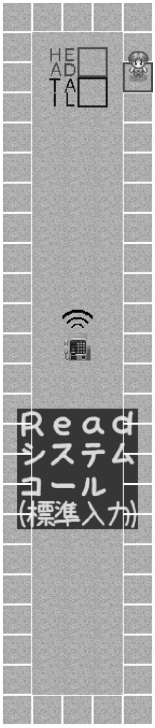
3.2 構成要素

Read システムコールは、以下の要素で実現されています。

- NPC

- 電話

- キーバッファ



- 先頭（Head）カーソル/末尾（Tail）カーソル

  - Head は赤色の四角、Tail は青色の四角です。（白黒の印刷ではどちらか分からないのでスクショは一枚だけですが。[*1]）

[*1] ちなみにコレは赤色の四角です。

3.3　動作の流れ

♣ 1. キー入力を待つ

Read システムコール担当の NPC の初期位置は、キーバッファの Head 位置です。そこに、入力されたキーを示す文字パネルが来るのを待ちます。

♣ 2. システムコールの電話を待つ

キーボードバッファの Head に文字パネルが存在することを確認すると、Read システムコール担当はシステムコールの呼び出しを待ちます。

♣ 3. 入力されたキーを取得する

システムコールが呼び出されると、Read システムコール担当は自分が立っている Head 位置の文字パネルを取得します。この際に、Head カーソルが自動的に 1 文字分移動します。

♣ 4. 電話に出て、キーを伝え、電話を切る

文字パネルを取得した後、Read システムコール担当は Read システムコールの電話の正面まで移動し、電話に出ます。そして、取得した文字パネルの文字を相手へ伝え、電話を切ります。

♣ 5. Head 位置へ移動し、再びキー入力を待つ（1 へ戻る）

電話を終えると、現在の Head 位置へ移動し、再びキー入力を待ちます。すなわち、1 へ戻ります。

【コラム】RPGOS の Read システムコールは同期型

Read システムコールも、Write システムコールと同様、「システムコールの電話を待つ」以外のフェーズでは、電話がかかってきても出ません。キー入力を待っている間に電話がかかってきても、「ありません」ということを応答するのではなく、電話に出ず放置します。そのため、システムコールを呼び出した側はキー入力があるまで待つこととなり、RPGOS の Read システムコールとしても、同期型（ブロッキング）という訳です。

3.4　現状の仕様や課題、小ネタなど

♣ キーバッファは謎技術

前章でも説明した通り、裏側の RPG 世界では、基本的に我々の世界でも存在するものを用いて「人力」で OS の機能を実現しています。ただ、このキーバッファだけは、今のところ、何が起きているのか説明できない謎技術になっています。RPG 世界の住人からみると、突然文字パネルが出現してくる領域といったものです。もっと我々の現実にもある何かでキーバッファを表現できないか考えているのですが、今のところ、良いものが思いついていません。

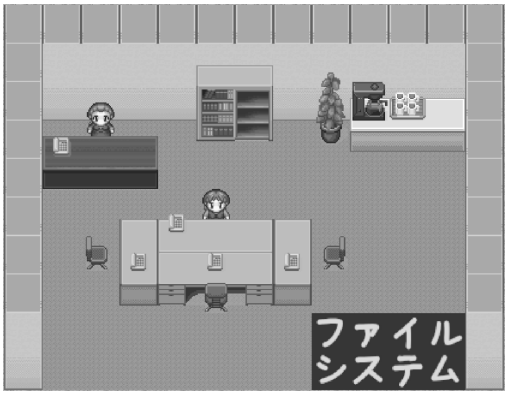
第 4 章
Open システムコール

この章では、「Open システムコール」について解説します。

4.1 どのようなシステムコールか？

パスで指定されたファイル/ディレクトリのファイルディスクリプタを返すシステムコールです。

なお、Open システムコールとファイルディスクリプタ（次章で解説）は、ファイルシステム（図 4.1）内にあります。



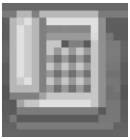
▲図 4.1: ファイルシステムの領域

4.2　構成要素

左上に居るのが Open システムコールです。以下の要素で実現されています。

- NPC

- 電話



なお、NPC の上に状態を示すアイコンが出ることがあります。アイコンの意味は以下の通りです。

Open システムコール担当の状態を示すアイコン

ファイルディスクリプタ担当を呼んでいる


4.3　動作の流れ

♣ 1. システムコールの電話を待つ

まず、Open システムコール担当の NPC は、自身の座席でシステムコールの電話がかかってくるのを待ちます。現状、Open システムコール担当は座席から移動することはありません。

♣ 2. 電話に出て、パスを聞く

電話がかかってきたら出て、電話相手から開きたいファイルあるいはディレクトリのパスを聞きます。その際、相手から聞いたパスが存在するか確認します。現状では、"/"（ルートディレクトリ）と"HELLO.TXT"のみが存在するディレクトリあるいはファイルとして扱われます。それ以外のパスが指定された場合は、エラー（"NO SUCH FILE OR DIRECTORY"）を返します。エラーの場合はここで電話を切ります。そのため、エラーの場合は 6 へ遷移することになります。

♣ 3. ファイルディスクリプタ担当 NPC を呼び出す

電話を切らず、そのまま、ファイルディスクリプタ担当の NPC を呼びます。

♣ 4. ファイルディスクリプタ担当 NPC へパスを伝える

ファイルディスクリプタ担当 NPC が反応したら、パスを伝えます。

♣ 5. 電話を再開し、ファイルディスクリプタの番号を伝える

電話口で相手へ、パスを伝えたファイルディスクリプタ担当の番号を伝えます。

♣ 6. 電話を切り、再び電話を待つ（1 へ戻る）

用件を終えたら電話を切ります。そして再びシステムコールの電話が来るのを待ちます。すなわち、1 へ戻ります。

4.4　現状の仕様や課題、小ネタなど

♣ Open システムコール担当は存在するパスを覚えている

現状、Open システムコール担当は、相手からパスを聞くと、本棚へ確認に行かずにそのパスが存在するか否かを判断します。正直な所、これはその流れを実装し忘れていたというものなのですが、一つの考え方としては、今のところ"/"と"HELLO.TXT"しか存在しないので、それくらいなら担当者本人が覚えている、あるいは手元にメモっていて本棚まで見に行く必要がない、ということが考えられます。

なんだかこれは、「人」で実現するように考えたからこそ自然に生まれたキャッシュ機構なような気がします。現実でこのようなことをしようとしたとしても、"/"と"HELLO.TXT"しかないくらいなら都度確認しには行かないだろうと思います。

他方、今後、ファイルやディレクトリが増えてきた場合、Open システムコール担

当が本棚まで確認に行くようになるのかというと、それもちょっと疑問があります。現状、Open システムコール担当は一人で、後ほど詳しく紹介するファイルディスクリプタ担当は複数人存在する事ができる構成になっています。このような構成では、ファイルアクセスが頻繁に行われるようになると Open システムコール担当がボトルネックになってきそうな気がします。そのような中で、Open の都度、Open システムコール担当が毎回席を立って本棚まで移動するのはちょっと避けたいです。例えば、やはり、頻繁にアクセスされるファイルやディレクトリについては手元にメモっておくとか、あるいは本棚の前に一人立っておいてもらって、Open システムコール担当が座席を立たず、その人に声で伝えて、パスが示すファイル/ディレクトリの有無を確認してもらう、とかが考えられます。

第 5 章
ファイルディスクリプタ

この章では、「ファイルディスクリプタ」について解説します。

5.1 これは何か？

「ファイルディスクリプタ（以降、FD と呼称します）」とは、Open されたファイルあるいはディレクトリに対する一意の識別子です。ファイル/ディレクトリに対応した（あるいはそれ用の）システムコールへ FD を指定して実行すると、システムコールの対象がそのファイル/ディレクトリになります。

RPGOS では以下のシステムコールが FD に対応しています。

Read システムコール

FD を指定すると、それがファイルを Open したものである想定で、そのファイルの内容を読み出して返します。現状、1 度のシステムコールで返されるのは 1 行です。最後の行を返した次のシステムコールでは""（空文字列）が返されます。ディレクトリを Open した FD を指定した場合の挙動は未定義です。

GetDEnts システムコール

FD を指定すると、それがディレクトリを Open したものである想定で、そのディレクトリ内のファイル/ディレクトリ名を返します。現状、1 度のシステムコールで返されるのは 1 つです。最後のファイル/ディレクトリ名を返した次のシステムコールでは"/"が返されます。FD を指定しなかった場合や、FD がファイルのものである場合の挙動は未定義です。

Close システムコール

FD を指定すると、Open していたファイル/ディレクトリを Close します。RPGOS としては、FD 担当の NPC が該当のファイル/ディレクトリを担当するのをやめることになります。

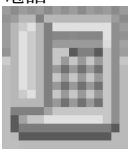
なお、一般的な UNIX 系 OS では、Read/Write システムコールは FD の指定が必須で、標準入力の FD が 0、標準出力の FD が 1[*1]という扱いになっているかと思います。RPGOS では、現状、「FD が指定されなかった場合、標準入出力を対象にする」という形になっています。

5.2　構成要素

ファイルシステム領域の右下に居るのがファイルディスクリプタです。以下の要素で実現されています。

- NPC

- 電話



- 本棚

[*1] 加えて、標準エラー出力の FD が 2 です。

なお、NPC の上に状態を示すアイコンが出ることがあります。各アイコンの意味は以下の通りです。

FD 担当の状態を示すアイコン

Open システムコールからの呼び出しに答えた	担当中（FD 使用中）
	

5.3 動作の流れ

♣ 1. Open システムコール担当からの呼び出しを待つ

FD 担当の NPC は、自席で Open システムコール担当からの呼び出しを待ちます。

♣ 2. Open システムコール担当からの呼び出しに応答する

Open システムコール担当から呼び出されると、FD 担当は反応を示し、パスを聞きます。なお、Open システムコール担当の章でも説明した通り、現状で対応しているのは、"/"（ルートディレクトリ）あるいは"HELLO.TXT"のいずれかです。

♣ 3. 本棚まで移動し、少しゴソゴソした後、自席へ戻る

パスを受け取った後、FD 担当は本棚まで移動し、数秒程ゴソゴソした後、自席へ戻ります。ゴソゴソしているのは、パスがディレクトリの場合は「ディレクトリ内のエントリ（ファイル名あるいはディレクトリ名）をメモっている」、あるいはパスがファイルの場合は「そのファイルを探して手に取っている」といったことを想定して、NPC の動きは実装しているのですが、実は内部的には何もしていません。後述しますが、ディレクトリやファイルの情報は、この FD 担当の NPC 内にハードコードされています。

♣ 4. 電話を待つ

自席へ戻った後、自身の FD 番号を指定したシステムコールの電話を待ちます。

♣ 5. 電話に出て、対応する

電話がかかってくると、FD 担当が応答します。そして、その時の状況に応じて対応します。現状では「状況に応じて対応」というのが、対応できる状況について完全にベタ書きされています。具体的には以下の通りです。

■ パスが"/"の場合

- Open してから（自身がそのファイル/ディレクトリを担当してから）1 回目の電話の時：
  - "HELLO.TXT"を伝えて電話を切り、次の電話を待つ（4 へ戻る）。
- Open してから 2 回目の電話の時：
  - ""を伝えて電話を切り、次の電話を待つ（4 へ戻る）。
- Open してから 3 回目の電話の時：
  - Close 処理へ進む（5 へ進む）

■ パスが"HELLO.TXT"の場合

- Open してから 1 回目の電話の時：
  - "HELLO WORLD~n"を伝えて電話を切り、次の電話を待つ（4 へ戻る）。
- Open してから 2 回目の電話の時：
  - "/"を伝えて電話を切り、次の電話を待つ（4 へ戻る）。
- Open してから 3 回目の電話の時：
  - Close 処理へ進む（5 へ進む）

なお、現状、RPGOS ではエスケープシーケンスを示す文字を"~"（チルダ）としています。そのため、"~n"は改行文字です。

このように、現状では「"/"（ルートディレクトリ）の中に何があるか」や「"HELLO.TXT"というファイルの中身」といった情報は、この FD 担当の NPC のコードにハードコードされています。本棚へ行くのはフリで、実際は記憶しているものを返しているような形です。[*2]

♣ 5. Close 処理

電話口の相手からのクローズ宣言を待って、電話を切ります。そして、該当のファイル/ディレクトリの担当をやめ、再び Open システムコールから呼ばれるのを待ち

[*2] 「本棚へ行ってゴソゴソしていたら思い出したので何もせず帰ってきた。」と考えられるかもしれません。

ます。すなわち、1 へ戻ります。

5.4　現状の仕様や課題、小ネタなど

♣ 複数ファイルを同時に開く場合

現状では、一つのファイル/ディレクトリが Open されていると、FD 担当が対応中のため、それ以上ファイル/ディレクトリを開くことができません。

これに関しては、座席と電話の数、FD 担当の人数を増やすことで対応する想定です。座席と電話は既に 4 つずつあります。FD 担当の人数さえ増やせば、最大 4 つまで同時にファイル/ディレクトリを開くことができます。

そして、これが RPGOS におけるファイルディスクリプタ数の上限です。

「ファイルディスクリプタ数の上限にひっかかってプロセスがエラー終了する」というのは、OS 周りのエンジニアではない方には分かり難い概念かと思いますが、このような形で可視化されていると、それも何となく分かっていただけるかと考えています。

第 6 章
シェル

この章では、「シェル」について解説します。

6.1 これは何か？

ユーザーのキー入力をコマンドとして解釈し、実行する、OS の UI です。そのために、これまで紹介してきたシステムコールを実行します。

6.2 構成要素

シェルは、以下の要素で実現されています。

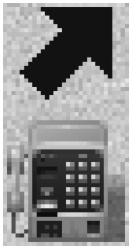
- NPC

- 電話

なお、現状、コマンド実行の際は、コマンドの処理を NPC 自らが行います。コマンドの処理がシェルにハードコードされている「シェル組み込みコマンド」のような状態です。

また、シェルの場合、電話の状態を示すアイコンとして以下が表示されます。

電話の状態を示すアイコン

通話中（電話をかけた側）


6.3　動作の流れ

♣ 1. プロンプトを表示する

シェル担当の NPC は、まずプロンプト（"$ "）を表示します。そのために、標準出力への Write システムコールを行います。

♣ 2. ユーザーからの文字入力を取得する

プロンプト表示後、標準入力に対する Read システムコールを行い、ユーザーからの文字入力を待ちます。そして取得した文字は以下のように処理されます。

- "~n"（改行）：入力文字列を空白で分割し、コマンドとして解析
- "~b"（バックスペース）：入力文字列の最後の 1 文字を削除
- その他の文字：入力文字列に追加

♣ 3. 取得した文字を出力する

取得した文字を出力するために、標準出力への Write システムコールを行います。その後の分岐は取得した文字により変わります。

■ 取得した文字が"~n"の場合

- 入力文字列をコマンドとして実行（4 へ進む）

■ 取得した文字が"~n"以外の場合

- ユーザーからの文字入力の取得へ戻る（2 へ戻る）

♣ 4. コマンドを実行する

改行が入力されると、入力文字列は空白で分割され、最初の要素がコマンド名として扱われます。コマンドの実行が終わると、シェル担当 NPC は再びプロンプトを表示する状態に戻ります。すなわち、1 へ戻ります。

6.4 各コマンドの処理の流れ

前述の通り、現状では、各コマンドの処理はシェル担当の NPC 自ら行います。現在実装されているコマンドとそれぞれの処理の流れは以下の通りです。

♣ ECHO：引数として与えられた文字列を表示

処理の流れ：

1. 与えられた文字列で標準出力への Write システムコールを行う
2. コマンド実行を終了する

♣ LS：カレントディレクトリの内容を表示

処理の流れ：

1. "/"を指定して Open システムコールを行う
2. 得られた FD を用いて GetDEnts システムコールを行う
3. GetDEnts から得られた文字列が"/"だった場合：
    1. 得られた FD を用いて Close システムコールを行う
    2. コマンド実行を終了する
4. 得られた文字列で標準出力への Write システムコールを行う
5. 再度 GetDEnts システムコールを行う（2 へ戻る）

♣ CAT：指定されたファイルの内容を表示

処理の流れ：

1. コマンドライン引数が指定されていない場合：
    1. 文字列"MISSING ARGUMENT~n"で標準出力への Write システムコールを行う
    2. コマンド実行を終了する
2. コマンドライン引数の文字列をパスとして Open システムコールを実行する
3. エラーが返された場合：
    1. 返されたエラーメッセージの末尾に"~n"を追加し、標準出力への Write システムコールを行う
    2. コマンド実行を終了する
4. 得られた FD を用いて Read システムコールを行う
5. 得られた FD を用いて Close システムコールを行う
6. Read 時に得られた文字列で標準出力への Write システムコールを行う
7. コマンド実行を終了する

♣ SHOWINSIDE：RPG 世界を表示

処理の流れ：

1. 内部状態を変更する（真っ黒のレイヤーを非表示にする、スクリーンの透過度を上げる）
2. コマンド実行を終了する

♣ HIDEINSIDE：RPG 世界を隠す

処理の流れ：

1. 内部状態を変更する（真っ黒のレイヤーを表示する、スクリーンの透過度を下げる）
2. コマンド実行を終了する

♣ NPCSLOW：NPC の動きを遅くする

処理の流れ：

1. 内部状態を変更する（NPC を遅いモードへ変更する）
2. コマンド実行を終了する

♣ NPCFAST：NPC の動きを速くする

処理の流れ：

1. 内部状態を変更する（NPC を速いモードへ変更する）
2. コマンド実行を終了する

6.5　現状の仕様や課題、小ネタなど

♣ 仮想ファイルシステムを実装したい

シェル上の操作感は Linux 的にしたいと考えています。現状では、「裏側の RPG 世界の表示/非表示」や「NPC のスピード設定」は独自のコマンドにより設定変更できる形になっていますが、Linux 的にするなら、このような OS の設定は「sysfs」や「procfs」といった仮想ファイルシステムで行えるようにしたいと考えています。これらはカーネルが管理しているデバイスやファイルシステム、その他カーネル自体の設定などを設定したり、現在の設定値を確認したりするためのインタフェースで、sysfs は `/sys` に、procfs は `/proc` にマウントされています。これらは、ユーザーからはそれぞれのパスの下にあるファイルとして見えていますが、実際にそのアクセスはファイルシステムへ行くのではなく、カーネル側で該当のファイルへのアクセスを検出すると、それが書き込みである場合は対応する設定を変更し、読み出しである場合は設定値を返します。これにより、ユーザーはファイルアクセスと同様の方法でカーネルの設定変更などを行うことができます。実際のファイルシステムではなく仮想的なものであるため「仮想ファイルシステム」と呼ばれます。例えば、`/sys/kernel/showinside` に `1` を書くと RPG 世界が表示されるようになり、`0` を書くと非表示になる、といったイメージです。ただ、このような操作はシェル上ではリダイレクト（`>`）で行いますが、そもそも現状ではまだシェルにリダイレクトの機能が無いです。そのようなこともあり、今今は独自コマンドという形にしていたりします。

【コラム】ゲームで Linux のシェルの基本操作を学ぶソフトにできるかも！

完全に妄想ですが、裏側の RPG 世界はゲームのような見た目をしているだけでなく、ゲーム的なイベントがあっても良いかなと思っています。例えば、突然、何かに洗脳された NPC がファイルシステム領域にある本棚のファイルを

次々に破壊してしまう、とかです。その場合、一刻も速くそのような NPC の動作を止める必要があるため、直ちに該当の NPC の ID を調べ、`kill` コマンドでシグナルを送って動作を停止させる[*1]、といった感じです。他にももっと面白く、RPG 世界に怪獣が現れて、標準出力の Write システムコール辺りを破壊してしまう、とかです。その場合、破壊されたところを復旧させつつ、当面の間はシェルで標準出力をする際は、まだ生き残っている別の Write システムコール領域の担当へ電話するように切り替える、といったことを行うなどです。

これらのイベントへの対応をシェルの UI 上で行うことで、ゲーム的に Linux のシェル操作を学ぶことができるのではないかなと、妄想しています。

[*1] ちなみに、kill コマンドが NPC へ SIGTERM や SIGKILL といったシグナルを送る操作は、「対象の NPC の停止」であって、「対象の NPC を殺す」ことでは無いと考えています。この RPG 世界において、NPC は設定されたプログラム（現状は完全にハードコードされていますが）にしたがって動く存在なので、PC における CPU あるいはスレッドに相当すると考えています。そのため、kill コマンドで停止系のシグナルを送った際に行うのは、CPU が実行するプロセスを止めることなので、NPC が現在行っている動作を止めることだと考えています。（あまり物騒な世界にしたくなくて頑張って考えた所でもあります（笑）。）

おわりに

ここまで読んでいただきありがとうございます！ 本書では、現在作成中の RPGOS を紹介しました。RPGOS は OS と言いつつも、PC や VM などではなく、RPG のゲームのような世界の上で動くというものです。これを OS と呼んで良いのかは微妙な所かとも思います。個人的には、「プログラムを実行できる CPU が人の姿をしている」というとても特殊なハードウェアを仮想的に実現している VM だと考えると、その上に「システムコール」という方法で機能が抽象化されていて、さらにその上でシェルが動いている、というのは「OS」と呼んでも良いんじゃないかな、という考えです。

いずれにせよ、本当に重要なのは、これが OS であるかどうかよりも「面白い」と思っていただけるかどうかかと思っているので、本書を読んで理解は難しくとも「何だか面白い」と思っていただければ幸いです。

なお、7 章の最後に「Linux のシェル操作を学べる」ことを応用例として挙げました。これはあくまでも一例で、「シェルの裏側の OS が RPG のゲーム的な世界で動く」というこの取り組みは、何か他にも色々と面白い発展へ繋げられそうな気がしています。（実装が全然追いついていないですが・・）

RPG のような世界で動く OS の本

2025 年 5 月 31 日　ver 1.0 （技術書典 18 新刊）

著　者　大神祐真
発行者　大神祐真
連絡先　yuma@ohgami.jp
　　http://yuma.ohgami.jp
　　@yohgami (https://x.com/yohgami)
印刷所　日光企画

(powered by Re:VIEW Starter)