本稿は Linux Japan 誌 2000 年 11 月号に掲載された記事に補筆修正したものです.

ジョイスティックで遊ぶ

この号が出るのは中秋の名月の頃ですが，原稿を執筆している今は厳しい残暑の夏です．丁度学校も休みですしたまにはゲームの話をしましょう．筆者は，はっきりいってゲームは好きです．もっとも，世界に冠たるゲーム機でのことで，普段は 娘と PS で RPG などを楽しんでいます．3DCG では専用チップを積んでいる PS に対抗できるはずがありません．しかし，ゲームの本質が変化しているわけでもなく，20 年位前のゲーセンで流行った単純なものがかなり楽しめます．仕事の合間には，あまりのめり込まずに済む程度のゲームが最適といっておきましょう (ところがどっこいハマッテてしまうのがゲームなんだよね (^^;).

「ゲーム全般に関しては，JG のあべさんが連載をしているしムックもでてるしなあ，それとは一味違う記事にせねば」と案を練りつつ，新宿のソフマップを物色していたところ，1000 円のジョイパッド (Justy JPD-108) が目に飛び込んできました.「おお，これだ！」というわけで，joystick(pad) ドライバーをカーネルに組み込んで，ゲーム環境を整備し (キーボードでは打ちゲーはつらいっすから)，Linux Game MASTER に名前を残せるような記録を打ちたてる準備をしようではないかというのが今回のテーマです.

図 1　トライコーポレション社製の安価な 2 軸 8 ボタンのジョイパッド Justy JPD-108：中央は連射ボタン，向こう側には L,R ボタンも付いています.

joystick driver

現在 joystick driver の保守・開発は，Vojtech Pavlik(vojtech@suse.cz) 氏が中心となって行なわれており，情報を

`http://www.suse.cz/development/joystick/`

より入手できます（リンクが切れています）．現在のバージョンは 1.2.15 です．なお，0.* のインターフェースとの互換も保たれていますから，0.* を前提にしているゲーム (Xsoldier 等) も，ちょっとした手直しで動くようです.

カーネル再構築

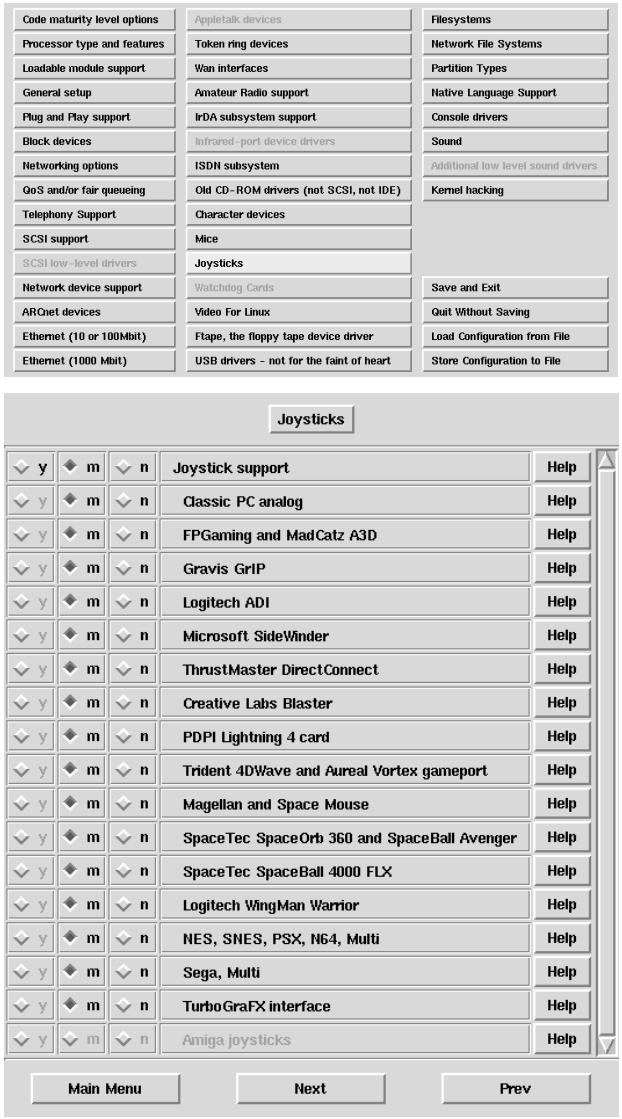


図 2　make xconfig によるカーネルの設定

ハード的には joystick は音源カードの MIDI(ゲーム) ポートに接続して使います．もちろん，デバイスドライバーを組み込まないと周辺機器は使えません．まずは，カーネルを再コンパイルしましょう．2.2 系ではソースに取り込まれており，

/usr/src/linux/drivers/char/joystick/joystick.txt を参考にして，すぐに構築にかかることができます．ご存知とは思いますが，一応手順を示します．それでは root になってください．まず古いモジュールのデリレクトリは別名で退避しておいた方が良いでしょうから /lib/modules に移動して

```
# mv 2.2.14  2.2.14.org
```

などとします．カーネルのバージョンは各自の環境にあわせて読み換えてください．続いて，/usr/src/linux に移動して，

```
$ make mrproper
$ make xconfig
```

します．メインメニューから joystick を選択し (図 2 上)，デバイスは全て module にしておきましょう (図 2 下)．Sound も組み込むのが常識でしょうね (joystick ドライバーは，Sound ドライバーとは独立していますが，ハードとしてのポート番号を知る必要があります)．設定を保存して，

```
$ make dep bZImage modules modules_install
$ depmod -a
$ lilo  (/etc/lilo.conf の確認を忘れずに)
```

を順に実行し，再起動してください．

デバイスファイル /dev/js*

忘れてはいけません，デバイスファイルも必要です．名前は /dev/js* です．もしこのデバイスファイルがない場合には

```
# mknod /dev/js0 c 15 0
# mknod /dev/js1 c 15 1
```

で作成してください．

ドライバーのロード

筆者のマシンにはオンボード ESS Solo-1 があって，Sound ドライバモジュール esssolo1 を modprobe でロードすると，

```
solo1:  joystick port at 0xe401
```

と出力されます (dmesg で確認)，他の音源ボードでも，この joystick(MIDI) ポート番号を探し出さなければなりません．

これからが本番で joystick.txt に従って，ドライバを選択します．このジョイパッドは汎用のドライバ joy-analog で認識できますから，

```
# modprobe joy-analog js_an=p0,m0,n0
```

として，必要なパラメータを渡してロードします．p0 はポート番号 (アドレス)，m0 は joystick 0 の型を指定するビットマスクです．n0 は joystick 1 のビットマスクです．p0 は先程得た情報から 0xe401 とし，joystick.txt にある表より，2 軸 8 ボタンを全て有効にするには m0 を 0xf0f3 とすれば良いことが判ります．2 軸 2 ボタンしか認識させないならば，m0 を 0x33 とします．

間違いなくロードされているか lsmod で確認してください．

```
# lsmod
Module                  Size  Used by
joy-analog              5192   0
joystick                5820   0  [joy-analog]
...
```

jstest

joystick のホームページで，joystick-1.2.15.tar.gz などの package が配布されており，テストツールが同梱されています．展開して jstest を以下のようにしてコンパイルします．

```
# make jstest
```

さっそくテストしましょう．

```
$ jstest --normal /dev/js0
Joystick (Analog 2-axis 8-button joystick ...
(略)...
Driver version is 1.2.15.
Testing ... (interrupt to exit)
Axes:  0:   0  1:   0 Buttons: 0:off 1:off (略)
```

2 つの軸と 8 つのボタンが認識されますから，パッド (X-Y 軸) やボタンを動かしてみてください．パッドに関しては数値 $-32767, 0, +32767$ が表示され，ボタンに関しては，状態 on off が表示されます．パッドではない本来の joystick はアナログデバイスですから，棒の傾きに応じた数値が入力されることでしょう．パッドは 3 ステートのスイッチとして使うことになります．TURBO(連射) ボタンを押すと on/off が高速に繰り返されます．

オプション --old により，バージョン 0.* の互換モードでテストできます．ボタンは 2 つしか認識しません．従来は数値に制限がなかったのですが，1.2.8 からは，0.* のモードに関しては 1 から 255 の範囲を戻すことになっているようです (従って，128 が中央値)．

```
$ jstest --old /dev/js0
(略)...
Axes: X:128 Y:128 Buttons: A:off B:off
```

ということで，無事ジョイパッドからの入力が確認できました．

calibration

パッドの位置に応じて入力される数値は各メーカーで同じではないようです．そこで，0〜1000 の範囲に正規化するためのツールが jscal です．アナログ値を使う場合には是非とも必要でしょう．が，パッドでは却って不安定になってしまいました．今回試したゲームなどは，アナログ値が必要ではありませんので，そんなツールがあるということだけを紹介しておきます．

ライブラリ libjsw

自然な事柄ですが，ジョイスチックを使うための汎用ライブラリが開発されています．Xship Wars の X11 化を開発している WolfPack のサイトをご覧ください [1]W[3]．GTK ベースの calibration ツールは jscal よりは使い易いです．

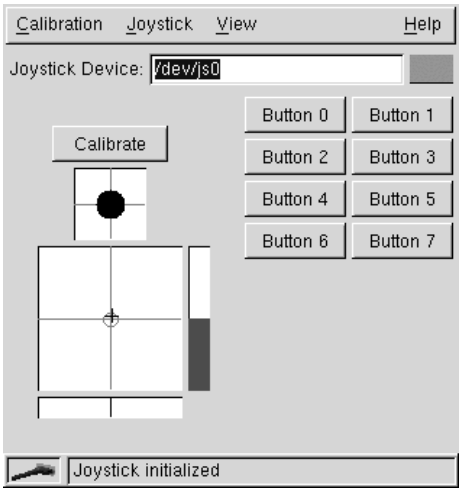


図 3　libjsw に同梱される jscalibrator

joystick 対応のゲーム

最初から joystick を想定しているゲームは少なく，JG では， Xsoldier だけのようです． バージョン 0.8 対応と書かれていますが，README.joystick の指示に従って無事コンパイルできます．ところが，joystick.c の最後の方で入力値から方向を算出するところに用いている値が，1.2.8 から変更されたものに合いません．そこで，その部分を次のように書き換えてからコンパイルしてみてください．0.* のモードを使うので，中央値の 128 を挟むようにしただけです．

```
if (js.x < 100) joymask |= Left;
if (js.x > 150) joymask |= Right;
if (js.y < 100) joymask |= Up;
if (js.y > 150) joymask |= Down;
```

joy2key

古い X11 上のほとんどのゲームが joystick に対応してません (最近のものは別です)．それでは，苦労してドライバーを組み込んだ意味がないことになります．ところが「蛇の道は蛇」，うまい抜け道があるのです．joy2key はゲーム制御を行うキー入力を joystick に置き換えるツールです．上下左右方向 (一般にはカーソルキー) と，2 つまでのキーに対応してます．現在，Peter Amstutz(tetron@student.umass.edu) 氏が保守しているようです [2]W[3]．以前からこのツールの存在自体は知っていたので，いつか試してみようとは思っていたのです．README には，例として xkobo が挙げられていますので，まずは一度その指示通りに使ってみるといいでしょう．結構面倒な設定が必要なので，$HOME/.joy2keyrc という設定保存ファイルを活用しましょう．xkobo と xtokkaetama の設定例を示します．-thresh は X-Y 軸 (パッド) の押された方向を判別する閾値です．この JPD-108 では ±32767 が両端ですから，README の手順に設定すると，±16383 を推奨してきます．判別されればいいのですから，区切りのよい 10000 としてみました．-X は X11 上のゲームという意味です．

```
COMMON
-thresh -10000 10000 -10000 10000
-X

START xkobo
-buttons n Shift_L m s Shift_L q Shft_L s
-axis Left Right Up Down

START xtokkaetama
-buttons KP_0 KP_5
-axis KP_4 KP_6 KP_8 KP_2
```

-axis ...　はパッドの押された向きをどのキーに対応させるかを記述しています．xkobo ではカーソルキーに対応させており判り易いですね．-buttons ...　はボタンとキーの対応を定義します．キーの名前は/usr/include/X11/keysymdef.h にある定義から，先頭の XK_ を除いたものです．xtokkaetama では，2 プレイヤーの方向キーは，キーパッドの (KP) の 4,6,8,2 ですから，このように設定します．対戦モードの場合，

キーボードだけだと狭くてやりにくいのですが，第 2 プレイヤーがキーボードから離れてとジョイパッドを使いますから，じっくり対戦できるようになります．

実は，配布のままでコンパイルしたものは -config オプションを使うと何故か Seg. fault してしまいました．引数の解析ルーチンに見当をつけてソースを調べた結果，check_config 関数に不具合をみつけました．434,465 行目にある，引数を格納する配列を開放している for ルーチンの終了条件がおかしいようです．ここを

```
for(; rcargc-1; rcargc--) から
for(; 0 == rcargc; rcargc--) に変更する
```

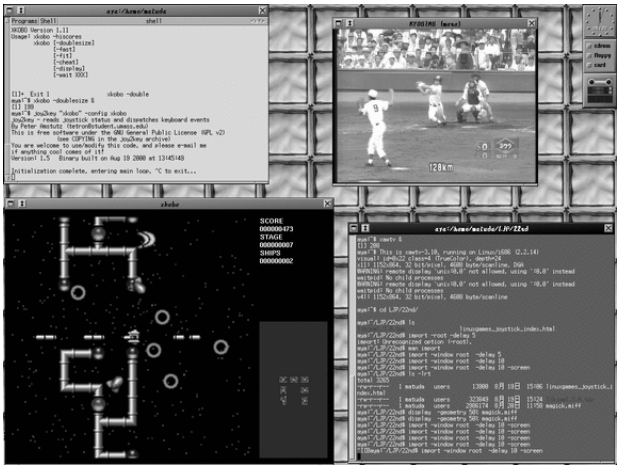
とすることで，とりあえず動くようになりました．

修正版をインストールして，いよいよ遊べます．具体的には，

```
$ xkobo &
$ joy2key "xkobo" -config xkobo
あるいは
$ xtokkaetama &
$ joy2key "XKaetama" -config xtokkaetama
```

です．二重引用符にはゲームの実際の起動名を指定しないといけません．xtokkaetama ではデフォルトが “XKaetama” となっていますので注意しましょう．

ジョイパッドを使っていることがサッパリわかりませんが，夏らしく高校野球を観戦しながら，xkobo をしている画面を示します．キーだと，特に小さいキーボードでカーソルが逆 T 字配列になっている場合，打ち間違いが結構ありますが，指に密着しているジョイパッドなら多少よそ見していても大丈夫です．



XInput では...

以上かなりのゲームがジョイパッドで遊べるようになります．ところが yamsweeper 等はマウスのみ操作可能ですから，joy2key も役立ちません．そこで，X サーバーの XInput 拡張を利用します．次のような記述を XF86Config に追加して，X を再起動しましょう．ルート画面上のカーソルをジョイパッドで動かせるようになります．

```
Section "Module"
  Load "xf86Jstk.so"
EndSection

Section "Xinput"
  SubSection "Joystick"
    Port "/dev/js0"
    DeviceName "Joystick"
    Timeout 1
    MinimumXPosition 1
    MaximumXPosition 255
    MinimumYPosition 1
    MaximumYPosition 255
    CenterX 128
    CenterY 128
    Delta 3
    AlwaysCore
  EndSubSection
EndSection
```

Delta を大きくすると移動が速くなりますが，位置決めが粗くなるので，3 くらいが適当な値と思います．それでも，マウスに比べると動きが鈍く，xmogura のようなアクション系や，移動量の大きいゲームは今一です．yamsweeper のように画面の片隅でチマチマ遊ぶのに向いているようです．

高得点は出たか？

いろいろなゲームをジョイパッドで操作してみました．テトリスなどの落ちゲーは，慣れてないせいもあって，微妙な操作ができず (余計に回転させてしまう)，キーボードの方が結果が良かったです．Xshisen などの大画面パズル系ではカーソルの動きが遅くて却ってイライラしました．結局，シューティング系の場合のみプラスの効果があって，椅子にもたれた楽な姿勢でプレイでき，疲れが少ないからでしょう，キーボードよりは得点があがりました．で，結論．ジョイパッドを使おうが使うまいが才能がないとダメなのだ．

参考文献

[1] Libjsw の公式ページ
http://wolfpack.twu.net/libjsw/

[2] キーボード入力をジョイスティックに振り替える
ツール joy2key
http://www-unix.oit.umass.edu/~tetron/joy2key