

# Setting up a Portable Intellivision Development Environment on Your Android Device

Written by Michael Hayes

<https://linktr.ee/intylab>

Date of Last Modification: January 8, 2023

---

(Note: don't let the length of this document intimidate you. It's designed to be easy-to-follow, rather than concise. You will only have to do this once.)

## Introduction

You have an Android device and a physical keyboard.

You now have some experience developing in IntyBASIC.

You would like to do future development using only your device and keyboard, so you can develop anywhere you're at in the cracks of time in your busy schedule.

You may not know anything about Linux and can't be bothered to "root" your device.

This document is for you.

## Disclaimer

I feel it is imperative to put this on the first page:

Neither I nor Midnight Blue International, LLC are responsible for anything bad that happens to you or your device for the use of any of the information in this text.

Neither I nor Midnight Blue International, LLC are responsible if you get fired from your job because you got caught writing games on company time.

Neither I nor Midnight Blue International, LLC are responsible if your life partner walks out on you because you're too busy making games anymore.

Standard data rates apply with your mobile carrier.

## Table of Contents

Introduction .....	1
History.....	3
Additional Notes .....	4
Step 0: Hardware requirements .....	5
Step 1: Install some apps. ....	6
Step 2: Establish contact with the outside world. ....	7
Step 3: Get some applications and write a script. ....	8
Step 4: Download and install IntyBASIC.....	9
Step 5: Download Advanced Console Component. ....	10
Step 5a: Download and install jzIntv/as1600.....	10
Step 6: Configure the Linux environment for graphics.....	12
Step 7: Install and configure some graphical applications. ....	13
Step 8: Additional actions .....	15
Trying “man” .....	15
Installing IntyColor .....	15
Step 9: Import your existing projects and create an archive.....	16
Step 10: Graphical Launchers .....	17
Step 10a: PIDEjL .....	17
Step 10b: PIDEGS .....	17
Appendix A: Future Startup and Shutdown Steps .....	22
Appendix B: Musings.....	23
Glossary.....	24
Acknowledgments.....	26

## History

### Fall 2018

When the book *Programming Games For Intellivision* was released, I bought a copy and read it right away, and with three days left to go in the 2018 IntyBASIC Programming Contest on AtariAge, I cranked out the game *Hunt The Wumpus*. I was so impressed with how easy IntyBASIC was to work with, that I came out of a 14-year hiatus after I wrote *SameGame & Robots* and the FUBAR prototype. The problem was that I couldn't make time in my schedule to sit down and work on new games, and carrying a laptop around was too impractical. I wanted to find a way to put a whole development environment onto a tablet, and I knew technology had improved to the point that it had to be possible.

IntyBASIC as well as jzIntv were available in Linux, and I knew that Android is Linux-based, so I just needed to find an app that would provide some kind of Linux environment, preferably with an unrooted device. Fortunately, I came across Termux, which at the time was available on Google Play. (Termux is still on Google Play, but that version no longer works.) With a bit of tinkering, I put together a text-based environment with a text editor, and some BASH scripts to compile and assemble IntyBASIC code.

### 2019

With a development environment now working, I could now improve upon it. I was previously using the app jzintv4droid to run games, but I knew I had to get jzIntv running from within a graphical environment to keep all development in one place. Joseph Zbiciak helped me every step of the way with this.

### 2020

To make the environment more palatable to newcomers, I wrote a couple GUI applications for playing and making games. These applications replaced all the original BASH scripts that you previously would have to write from scratch, and adapt to your own projects.

In July, jzIntv had a major update to support SDL2 instead of SDL1. I made the appropriate changes to the environment so it continued to work with the latest version of jzIntv. Later on, I also added IntyColor.

In November, I made my first attempt to create a video to demonstrate installing this environment onto a new device. During the recording, I discovered to my horror that the editor "emacs" no longer works.

### 2021-2022

One full-blown desktop environment was now fully available: XFCE4. I scrapped a lot of my earlier notes and replaced them with notes on installing this desktop environment. Between that and no longer needing to write your own BASH scripts, this document was reduced to roughly half its original size.

Android OS 10 was released, followed by versions 11 and 12. Significant changes were made to Termux behind the scenes as a result. I released new documentation along the way to keep up with the changes.

### 2023

I continue to watch for changes that affect installation and running of this environment. For example, with Android 11, there is a different way to access your MicroSD Card's filesystem.

I integrated code changes into jzIntv to allow for assembling code for the Advanced Console Component.

## Additional Notes

- Words that are underlined have a Glossary entry. Use the Glossary if you're having trouble understanding the context. Use of words you're probably not familiar with is unavoidable. Continuing past words you don't fully understand is a sure way to lose interest and walk away.
- Whenever there is text to enter, either within a text file or as a command, ...  
**it will look like this.**  
 Be sure to enter the text exactly as it appears, including capitalization. Pay careful attention to spacing as well.
  - I added spacing between the two following lines, so you know where line breaks are at.  
 Where there are multiple lines, ...  
**they will look ...**  
**like this.**
  - If a single line of text is too long to fit into a single line, it wraps around onto the next line without a space.  
**Here is an example of text which does not fit within a single line.**
  - Sometimes, you will have to make substitutions appropriate for your project or environment. Substitution text is colored in amber ...  
**like this.**
- I made a few files available as downloads on a page on my website, <http://www.midnightblueinternational.com/pide.php> for your convenience. Be sure to read the instructions on that page if you choose to download the files.

## Step 0: Hardware requirements

The first thing we need to do is make sure you are able to run this environment on your device.

You will need:

- Android OS 7 or higher
  - To find out what version of Android OS you have:
    - Swipe down from the top of the screen.
    - Click or tap on the gear icon which should be in the top right corner.
    - Scroll all the way to the bottom.
    - Click or tap on the item “About phone” or “About tablet”.
    - Look for the item “Software information” and click/tap on that.
    - “Android version” should be near the top.
- Approximately 2 GB of internal storage space, and at least 3 GB of RAM (4 GB is recommended)
  - To find out how much space and memory your device has:
    - Click/tap on the Back button to return to the Settings main menu.
    - Look for the item “Battery and device care” which should be about a half dozen items up from the bottom of the list. Click/tap on that item.
    - Below “Battery”, you will see how much storage is available.
    - Just below that is “Memory”. What’s important is how much RAM you have altogether, not how much is available at the moment.
- A physical keyboard for your device
  - This is not optional. There will be no on-screen keyboard once we have the desktop running. Even if it was available, you certainly don’t want to write code with it.
- I strongly recommend a mouse, but it is not required.
- I recommend a MicroSD card as well, but that is also not required.

I’ve installed this environment on a couple 10<sup>th</sup>-generation Samsung phones without incident.

As for tablets, the Samsung Tab A might not be sufficient, because Google Play reports that the app [VNC Viewer](#), which we will be using, no longer supports the Tab A. That means you’ll have to find another [VNC](#) client app and configure it yourself.

I am in the process of installing this environment on a Samsung Tab S6, and am trying again to create a demonstration video. Chances are, you’re reading this paragraph through that video. The Tab S6 should be fine.

Any devices newer than the above ought to be fine as well.

## Step 1: Install some apps.

We will be installing apps, both through Google Play, and using an alternate method. I will help you along the way.

- A. Go to <https://www.f-droid.org> and click/tap on the “Download F-Droid” button. Your web browser might give you a warning about installing apps from unknown sources. Bypass the warning and allow the download.
- B. When the download is finished, you should be prompted to install F-Droid. If not, find the .apk file in your Downloads folder and run it manually.
- C. Your device will probably also give you a warning about trying to install an app from an unknown source. Make the necessary changes to allow it to install.
- D. When it is finished installing, you should be prompted to Open/Launch/Run it. Do that now.
- E. You should see a message about “updating repositories”. Wait until it is finished.
- F. Tap/click on the Search button (a magnifying glass symbol with a round green button in the bottom right corner, just above the selection bar across the bottom) and type “Termux”.
- G. Tap/click the Download icon and install it when finished downloading. Don’t worry about launching it just yet.
- H. Download and install Termux:API in the same manner.
- I. If you want Termux:Styling as well, feel free to download and install that too.
- J. Go to Google Play and install “VNC Viewer – Remote Desktop”. No need to open it once it’s installed, just yet.

## Step 2: Establish contact with the outside world.

This is necessary for future steps. Because we're creating a [Linux](#) environment without [rooting](#), we'll perform a few steps so we can import files into the "[Linux](#) filesystem" and export files from here.

- A. Open Termux now. If you need to resize the text, you can zoom in & out the same way as with Google Maps: "pinch" or "stretch" the display.
- B. The first thing you should do is enter the command  
`pkg update && pkg upgrade`  
 to download any package updates. Future steps will not work until you do.
- C. You might get a Yes/No prompt about additional space usage. Answer Yes.
- D. Along the way, you will probably get additional Yes/No prompts. Answer No to these.
- E. When the package updates are finished, enter the command  
`pkg install termux-api`
- F. Enter the command  
`termux-setup-storage`  
 Your device will ask you to grant Storage permission to Termux at this point. Tap/click "Allow".
- G. So, what did this do? Try entering the "LiSt" command  
`ls`  
 and you will notice there is now a folder called "storage" which didn't exist before. You know it's a folder because Termux colors it violet by default.
- H. Go into this folder with the "Change Directory" command:  
`cd storage`  
 and notice "~/storage" appears in dark-green text in front of the "\$" prompt. That is your current directory.
- I. Enter the "LiSt" command  
`ls`  
 again. You will see there are several [symbolic links](#), which Termux colors cyan.

The one link called "shared" points to your device's Internal Storage. From the [Linux](#) filesystem, you can now import/export files to/from the Download folder within your device's Internal Storage via "storage/downloads". You can also update and delete files out there.

- J. Now return to your Home folder with this command:  
`cd`
- K. If you do not have a MicroSD Card, skip ahead to Step 3.  
 To find out how to access your MicroSD Card, enter the command  
`df`  
 At the end of the last line, you'll see something like this:  
`/storage/DEAD-BEEF`  
 If you read the Additional Notes earlier, you'll recall that text colored in amber varies.
- L. Make a [symbolic link](#) to the MicroSD Card's filesystem with this command:  
`ln -s /storage/DEAD-BEEF sdcard`

### Step 3: Get some applications and write a script.

Relax; I'll get you up and running in the graphical environment as soon as possible.

- A. Enter these commands:

```
pkg install man
```

```
pkg install x11-repo
```

```
pkg install xfce4
```

XFCE4 is a big one! With it, many other packages will be installed at the same time. Keep an eye out for any packages that fail to install. There are a couple other choices for desktops available now, but neither of them are ready for this kind of environment.

If your internet connection is fast enough, all the packages listed should have installed with no trouble. If any packages failed to download, use "pkg install" to try again to install them.

- B. Enter these commands:

```
pkg install xfce4-terminal
```

```
pkg install xorg-x11-fonts
```

```
nano .bash_profile
```

- C. nano is a text editor readily available with Termux. We'll install another one for the graphical environment once it's set up. For now, enter this text:

```
export PATH=.:$PATH
```

```
export DISPLAY=":1"
```

```
#export LD_LIBRARY_PATH=$PREFIX/lib/g14es
```

```
alias pkgup="pkg update && pkg upgrade"
```

- D. To save and exit, press Ctrl+X. Press 'y' to answer the "Save modified buffer" prompt. Press Enter to write to the file whose name we specified earlier.

- E. Create another file with this command:

```
nano x.sh
```

Enter these three lines:

```
vncserver -localhost -geometry 1920x1080
```

```
xfce4-session
```

```
vncserver -kill :1
```

Notice I colored the resolution in amber. If you happen to know your device's screen resolution, change this as you desire.

- F. Save and close as before.

- G. We are going to use the "chmod" command to make the x.sh file executable.

Enter this command:

```
chmod +x x.sh
```

- H. Now to create another symbolic link. Enter this command:

```
ln -s x.sh x
```



## Step 4: Download and install IntyBASIC.

We are going to get the latest version of IntyBASIC (currently 1.4.2) right from the source and put it directly into our [Linux](#) filesystem.

If you are new to [Linux](#), then get used to the idea of having to compile the source code for software packages and not having pre-compiled binaries (executable files). If a software package is available as an installable package, count it a blessing. Fortunately, I created this document to handhold you through this process.

- A. Enter these commands:

```
pkg install git
```

```
pkg install proot
```

```
pkg install wget
```

```
git clone https://github.com/nanochess/IntyBASIC
```

You now have a folder called “IntyBASIC” with IntyBASIC in it. In case you don’t know, file and folder names in [Linux](#) are case-sensitive. “IntyBASIC” and “intybasic” are not the same, for example, and both can exist within the same folder.

- B. Enter these commands:

```
cd IntyBASIC/intybasic
```

```
clang++ IntyBASIC.cpp code.cpp microcode.cpp node.cpp -o  
intybasic_termux
```

- The second command is wrapped around into two lines, since it doesn’t all fit into one line. Make sure there’s a space between “-o” and “intybasic\_termux”.
- You might get a warning here, but it should successfully create the executable file “intybasic\_termux” (no filename extension), which is what you’ll be using.

- C. When the “\$” prompt appears again, you know compilation is finished. Enter these commands to CoPy the appropriate files to the Home folder and return there:

```
cp intybasic_* ~
```

```
cd
```

```
chmod -x intybasic_*.asm
```

- D. Now close Termux. When we open it again, the changes we made earlier will take effect:

```
exit
```

## Step 5: Download Advanced Console Component.

ACC is a new add-on module for Intellivision. There are code modifications to allow you to assemble ACC code with as1600. Because we're going to be compiling jzIntv/as1600 from scratch anyhow, it is an easy thing to integrate this mod into the environment.

- Open your device's web browser and go to: <https://forums.atariage.com/topic/266459-birds-nest>
- You are at the beginning of the thread "Bird's nest" in the AtariAge Intellivision Forums. Navigate to the last page.
- Scroll up from the bottom of the page to find the latest download of "Advanced Console Component". As of right now, it's "Advanced Console Component 20221217.zip".
- Click/tap on the grey box to download this file.

## Step 5a: Download and install jzIntv/as1600.

jzIntv is the actual emulator and provides graphics and sound. I want to make sure this successfully compiles onto your device.

- Go to: <http://spatula-city.org/~im14u2c/intv/>
- Look at the right margin of the page where the downloads are, and then click/tap on the link that ends in "-src.zip". See the illustration to the right.
- Reopen Termux and enter these commands:

```
mkdir ACC
```

```
cp storage/downloads/Advanced\ Console\ Component\ 20220217.zip  
ACC/
```

```
cd ACC
```

```
unzip A*.zip
```

```
cd
```

```
cp storage/downloads/jzintv-20200712-src.zip .
```

```
unzip jzintv-20200712-src.zip
```

```
pkg install xorgproto
```

```
pkg install sdl2
```

```
pkg install gl4es
```

```
cd jzintv-20200712-src/src
```

```
cp ~/ACC/jzintv_mod/asm/*.asm
```

```
cp ~/ACC/jzintv_mod/imasm/*.asm
```

```
nano Makefile.termux_sdl2
```

**RECOMMENDED: Current Stable Dev Version**

- **New! SDL2 builds:** [\(Download SDL2\)](#)
  - [jzintv-20200712-win32-sdl2.zip](#)  
Windows 32-bit binaries.
  - [jzintv-20200712-osx-sdl2.zip](#)  
MacOS X 64-bit x86-64 binaries. (Should work w
  - [jzintv-20200712-linux-x86-64-sdl2.zip](#)  
Linux 64-bit x86-64 binaries. *Untested. Recomm*
  - [jzintv-20200712-linux-rpi-sdl2.zip](#)  
Raspberry Pi Raspbian Linux 32-bit binaries. *Unte*
- **SDL1 builds:** [\(Download SDL1\)](#)
  - [jzintv-20200712-win32-sdl1.zip](#)  
Windows 32-bit binaries.
  - [jzintv-20200712-osx-sdl1.zip](#)  
MacOS X 64-bit x86-64 binaries. (Should work w
  - [jzintv-20200712-linux-x86-64-sdl1.zip](#)  
Linux 64-bit x86-64 binaries.
  - [jzintv-20200712-linux-rpi-sdl1.zip](#)  
Raspberry Pi Raspbian Linux 32-bit binaries. *Unte*

• [jzintv-20200712-src.zip](#)  
jzIntv-20200712 source archive

**OLD, NOT RECOMMENDED: jzIntv 1.0 Beta 4**

It's necessary for us to tweak the Makefile before we attempt compilation. With these code modifications, there are going to be numerous compiler warnings. The author of jzIntv, Joseph Zbiciak, maintains a very high programming standard, and so he added a line to the Makefile to disallow compilation with warnings.


- D. You are now in the Nano text editor once again.
  - I. Press Ctrl+/, (the forward slash key, next to right Shift), type "38", and press Enter.
  - II. You're now at Line 38, which contains the text  
**WARN += -Werror**  
 "Comment out" this line by prefacing it with a hash symbol:  
**#WARN += -Werror**
  - III. Save and close.
- E. It's time to compile jzIntv now! Enter the command:  
**make -f Makefile.termux\_sd12**

I had a *lot* of help from Joe Zbiciak to get jzIntv working originally! It's going to take a few minutes to build, and text is going to flash all over your display. Try to avoid letting your device open a screensaver or go to sleep while it's compiling.

- F. There's a chance you'll get a Segmentation Fault message and an error code during the final Linking phase, if your device doesn't have a lot of RAM. If that happens, you'll have to make another small modification to the Makefile and try again. If compilation is successful, skip to Step 5G.
  - I. Once again, enter the command  
**nano Makefile.termux\_sd12**
  - II. Press Ctrl+/, type "52", and press Enter.
  - III. You're now at Line 52, which contains the text  
**LT0 = -flto**  
 "Comment out" this line by prefacing it with a hash symbol:  
**#LT0 = -flto**
  - IV. Save and close.
  - V. Try again to compile by entering the commands  
**make clean**  
**make -f Makefile.termux\_sd12**
- G. Once compilation is successful, you'll have a "\$" prompt again. Enter these commands:  
**cd ../bin**  
**cp as1600 ~**  
**cp jzintv ~**  
**cd**

## Step 6: Configure the Linux environment for graphics.

As with the other steps, you only need to do this once.

- A. Install the VNC server with this command:  
`pkg install tigervnc`
- B. Launch the X script, also from Step 3:  

- C. Create a password and enter it a second time to verify it. Length must be between 6 and 8 characters. No characters will echo as you type. There is no need to make a read-only password.
- D. Disregard the message from xauth that the file .Xauthority does not (yet) exist. For now, keep Termux running, and return to your Home Screen and open the VNC Viewer app.
- E. Tap/click on the “Next” button until you get to the end of the overview screens, and then tap/click on “Get Started”.
- F. You do not have to create an account. Just hide the menu pane on the left and tap/click on the green “+” icon to set up a New Connection.
- G. For the Address, enter  
`127.0.0.1:5901`  
 You can enter anything you want for the Name. I recommend “Termux”. Tap/click “Create”.
- H. A pane will appear on the right, where you can change the connection settings. I suggest you change the Picture quality to “High” unless you want the desktop to look like barf. It’s up to you whether to disable “Update desktop preview”.
- I. Tap/click the green “Connect” button at the bottom of the pane.
- J. You will get a warning about an Unencrypted connection. Disable “Warn me every time” and tap/click “OK”.
- K. Enter the password you just created at Step 6E. I suggest you enable “Remember password”. Tap/click “Continue”.
- L. It’s up to you whether to view the “mouse gestures” if you’re not using a mouse. You should see your desktop now.
- M. Tap/click the thumbtack icon to hide that white pane covering up the top of the screen. You need to be able to see your full display, especially at the top.

## Step 7: Install and configure some graphical applications.

Welcome to your new Desktop! Let's get acquainted with the Terminal Emulator and fetch some graphical packages now.

- A. The "Start Menu", if you're familiar with that term, is in the top left corner of the screen. Move the mouse pointer there and click on the "Applications" button.
- B. In the drop-down menu that appears, select "Terminal Emulator" at the top.
- C. Within the Terminal window, select "Edit" and "Preferences".
- D. Click on the "Appearance" tab.
- E. Look for "Default geometry" and change the columns from 80 to 160.
- F. Any other changes are up to you. I usually go with 35 rows (or more, depending on your device resolution), change the font size to 14, and make the background translucent. Click "Close" when you're done.
- G. Close the Terminal window and open it again, so the geometry changes will take effect.
- H. Within this Terminal window, we're going to install more packages. First is your choice of graphical text editor. I suggest you install geany, unless you're already familiar with the vim interface *and* your keyboard has an Escape key (early Android keyboards don't have an Escape key, and some don't even have Function keys). Otherwise, go with vim-gtk. Unfortunately, emacs-x no longer works.  

```
pkg install geany
```
- I. Now for the rest of the packages.  

```
pkg install netsurf
```

```
pkg install galculator
```

```
pkg install mtpaint
```

```
pkg install feh
```

```
pkg install xpdf
```

```
pkg install python-tkinter
```
- J. Click on the "Applications" button again, and you'll notice the list is now a little longer. For now, select "Web Browser".
- K. The window that pops up will ask you to choose an application. Select "Other" from the drop-down list.
- L. Type "netsurf", press Enter, and then click "OK".
- M. From now on, opening "Web Browser" will open netsurf. Close it for now.
- N. If you look at the "Applications" list, you'll discover galculator within Accessories, and mtpaint within Graphics. As for your text editor, geany appears within Development, and vim and gvim appear within Accessories, depending on which text editor you installed. If you installed geany, then open that from the menu; if you installed vim, then type "evim" from the Terminal window.

Now let's try changing the color scheme.

For geany:

- A. Using your device's web browser (not netsurf), go to <https://www.geany.org/download/themes/>
- B. Shop around for a theme you like.
- C. How to download the configuration file depends on the browser you're using (I'm using Microsoft Edge). Long-press the "Download" link (do not use your mouse) and select "Download link" from the pop-up window. Pay attention to whether or not your browser appends a ".txt" filename extension.
- D. Return to VNC Viewer.
- E. From the Terminal window, enter the command:  

```
cp storage/downloads/bespin.conf.txt
../usr/share/geany/colorschemes/bespin.conf
```

Note this command is one line, with a space before the two periods. Also note we're simultaneously renaming it without the .txt extension while we copy it into the folder where geany fetches all the available configuration files.
- F. In the geany window, select "View" and "Change Color Scheme" from the Menu Bar.
- G. The theme you downloaded should appear in the list beneath "Default" and "Alternate". Select that theme and click "Close".
- H. Another change I recommend is to not reopen all the same files in subsequent sessions.
  - I. From the Menu Bar, select "Edit" and "Preferences".
  - II. Click/tap on "General" and uncheck the first box under the Startup group: "Load files from the last session".
  - III. Click/tap "OK".

For vim:

- A. From the Terminal window, enter the command  

```
evim .vimrc
```
- B. Enter these lines:  

```
set nowrap
colorscheme industry
syntax on
if has ("gui_running")
    set columns=160 lines=35 guifont=Monospace\ 14
endif
```
- C. Save and close.

## Step 8: Additional actions

### Trying “man”

This is one of the first packages we installed. You might be advised at some point to “read the [man page](#) for more information” or something like that.

To read a [man page](#), type “man” followed by the software package. For example, to read the [man page](#) for feh, type “man feh”. Try it for man itself too: “man man”.

Use the spacebar to read the next page, or press Enter to scroll one line at a time. Press ‘q’ to quit.

### Installing IntyColor

If you’re familiar with IntyColor already and want to import it into this environment, feel free. I won’t go into detail about its usage here since this is not a programming manual. Installation is like that for IntyBASIC at Step 3.

From the Terminal window, enter these commands:

```
git clone https://github.com/nanochess/IntyColor
```

```
cd IntyColor/IntyColor
```

```
clang IntyColor.c -o intycolor_termux
```

```
cp intycolor_termux ~
```

```
cd
```

For help in using IntyColor (there is no [man page](#)), simply run the command:

```
intycolor_termux
```

## Step 9: Import your existing projects and create an archive.

Now that you're up and running in this environment, it's time to copy your project files from whatever device you were using before.

My method has been to copy all the folders and files onto a USB drive that has a USB connector as well as a Micro USB or Type-C connector, so you can connect to a PC as well as your mobile device. How you get the files onto your device is up to you. You can also use cloud storage if your device has access to an account such as Dropbox, Google Drive, or Microsoft OneDrive.

- A. Using your device's file explorer app (usually called My Files, but it varies across devices), first locate your projects, either on a USB drive or cloud storage.
- B. Create a subfolder called Archive, either from your MicroSD root folder or your Internal Storage's Downloads folder. Go into this new Archive folder.
- C. Create another subfolder called intyBASIC and go into this folder.
- D. I assume each of your projects is contained within its own folder. Copy all your project folders into this intyBASIC folder. Later, we will put backups of all source and target files here.
- E. Return to VNC Viewer. From the Terminal Emulator, enter these commands:

```
mkdir projects
```

```
cd projects
```

- F. Use the mkdir command again to "MaKe a DIRectory" for each project you are importing. The name does not matter, because we will have references to both the folder name here, where you will be working, and the folder in the Archive.
- G. Use the cd command to go into each subfolder, and use the command to copy all your files into the subfolder. If you *are not* using a MicroSD Card, this is the command:

```
cp ~/storage/downloads/Archive/intyBASIC/FUBAR/*.* .
```

If you *are* using a MicroSD Card, the symbolic link we created in Step 2 should work. Use this command instead:

```
cp ~/sdcard/Archive/intyBASIC/FUBAR/*.* .
```

- H. You don't have to do this, but since these files aren't going to be executable, let's eliminate Execute permissions with the "chmod" command:

```
chmod -x *.* .
```

- I. Repeat steps 9F to 9H for your other projects.
- J. Return to your Home folder with the command

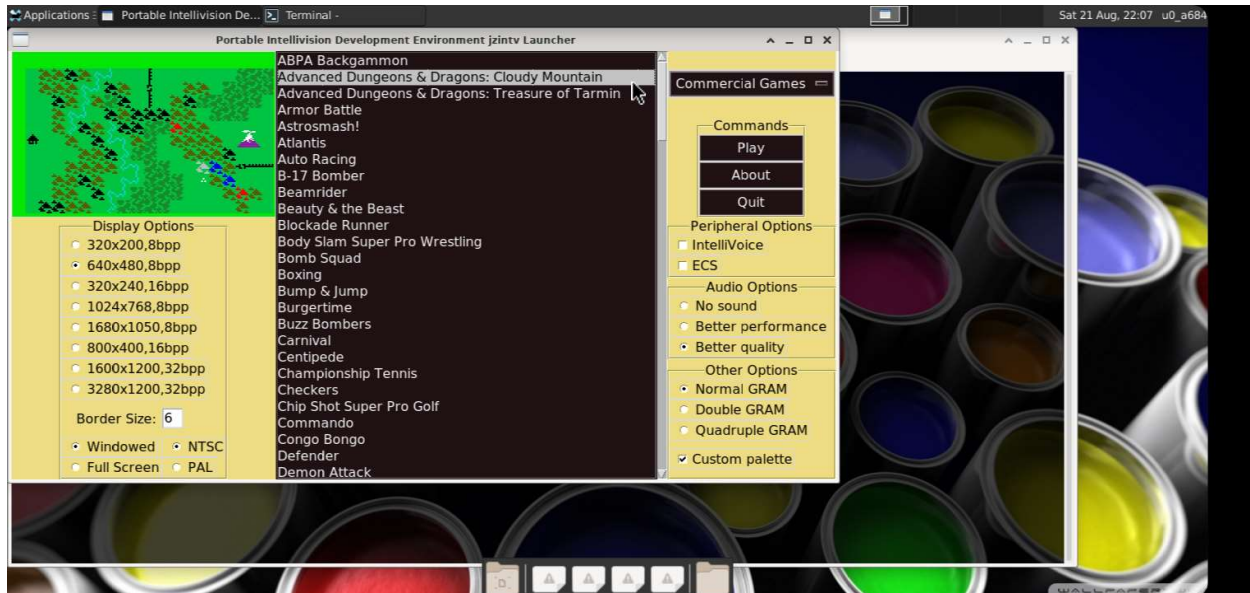
```
cd
```

- K. Finally, get your exec.bin, grom.bin, and ecs.bin files into your Home folder. Make sure they're lower-case. If you have the Tutorvision gromintv.bin and wbexec.bin files, import those as well.



## Step 10: Graphical Launchers (10a: PIDEjL)

There are two programs I wrote to make it easy either to play Intellivision games or to make them. First, we'll look at playing games.



- A. Go to <http://www.midnightblueinternational.com/pide.php> and download pidejl.zip (not pidejlw.zip; that's the Windows port). Again, use your device's web browser, not netsurf.
- B. Return to VNC Viewer and run these commands: (You should be in the Home folder, per step 9K)
 

```
cp storage/downloads/pidejl.zip .
```

```
unzip pidejl.zip
```

```
chmod +x pidejl/*.sh
```

```
chmod +x pidejl/*.py
```

```
ln -s pidejl/play.sh play
```

```
ln -s pidejl/make.sh make
```
- C. Using your text editor, open the file ~/pidejl/config.py
  - I. Do not delete any lines. Just make changes to the existing lines.
  - II. Change the paths as appropriate.
    - a. Make sure there's a slash at the end of the path strings.
    - b. If you're using your MicroSD Card in Android OS 11 or later, remember the slash at the beginning.
    - c. Since jzIntv is copied to the Home folder, the path to jzIntv shall remain an empty string.
  - III. For the defaults of the graphical options, there are comments just above each line to guide you in case you make any changes.
  - IV. The global options are parameters that will probably remain the same for all games. They can be overridden by individual game parameters. One example is the keyboard hack file for Vectron, which I'll cover shortly.

V. Now it's time to modify the Game List.

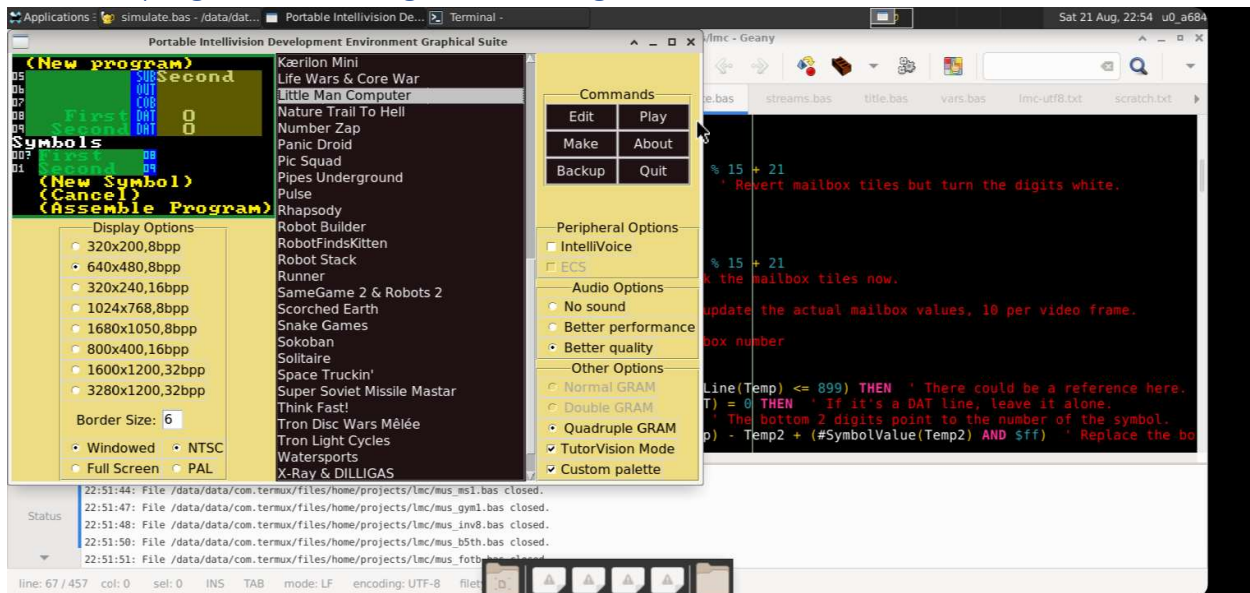
- a. The first element is the name of each game as it appears in the list. The games will appear in the order they are listed.
- b. The second element is the ROM image filename. The path to the ROM image files is already listed above.
- c. The third element is the default option for ECS. When you select a game, the ECS check button will be checked by default if it's set to 1. For all other games, 0 is used to uncheck the ECS button by default.
- d. The fourth element is the default option for IntelliVoice. 1 checks the IntelliVoice checkbox by default, and 0 unchecks it by default.
- e. The fifth element is for any future games that use expanded GRAM. 0 is the default for Normal GRAM, 1 is for games that use Double GRAM (available in modified Intellivision-II consoles), and 2 is for games that use Quadruple GRAM (available in late INTV consoles like the World Book Tutorvision). None of the original games used additional GRAM, and if you check one of the radio buttons to add more GRAM than the game calls for, you'll get a warning that enabling extra GRAM can cause graphical issues with certain games. At this time of writing, the only programs that utilize Quadruple GRAM are StudioVision, an RCA Studio II emulator for Intellivision, and Little Man Computer, a free program I wrote myself that is intended for Tutorvision.
- f. The sixth element is for any game-specific parameters you want. I added one for USCF Chess to disable speed throttling ("--ratecontrol=0"), to add default virtual tape filenames for three of the ECS games, and to use an alternate keyboard hackfile for Vectron, which will override the default keyboard hackfile. You should populate this element if you add indie games that use JLP.
- g. New as of April 2021 is a feature to categorize games into groups. A group list appears above the Command Buttons. There are two additional lines after the GameList for the groups. Follow the example to create additional groups.

VI. Save and close.

- D. To launch PIDEjL, enter the command **play**
- E. Click on a game title. The "Play" button will become selectable. If you have a screenshot for that game, it will appear in the preview window in the corner. The ECS and IntelliVoice checkboxes will check or uncheck themselves appropriately, and the appropriate GRAM Radio button will select itself as well.
- F. You can make changes to any of the options you like. Full Screen doesn't work in this environment, but I am leaving it enabled.
- G. Click on Play to play the game. The box and overlay images will appear in they are available. I'll explain the key bindings in a moment.
- H. When you quit jzIntv, the launcher window will reappear. I haven't gone over it yet, but I have the Quit function bound to Open Square Bracket **[** next to P.
- I. Click on the Quit button when you're done.

## Step 10b: PIDEGS

Now, the program for making Intellivision games.



At the bottom of `pidejl/config.py` is the development-related section. We're going to make a few changes here as well, so open that file again in your editor.

- A. Change the paths here if necessary.
- B. Change the editor if you're not using geany.
- C. Now for the ProjectList. I have a couple fictitious project examples here: my games FUBAR and X-Ray & DILLIGAS. You need to have at least two projects in the list.
  - I. The first element is the game title. It will appear this way in the list and will become the title in the ROM header. Ex: "FUBAR".
    - a. You'll notice in the illustration above that there are Unicode characters to display the titles "Kærilon Mini" and "Tron Disc Wars Mêlée". I just put those in for illustration purposes. The Intellivision does not have diacritics available natively, so I can't use those for my final ROM titles.
  - II. Second is the subfolder within the projectpath. If the game FUBAR resides within "projects/fubar", you would enter "fubar/" here. Don't forget the slash at the end of the folder name!
  - III. The third element is a sublist containing the files that you will open in your editor. Usually it will be ("\*.bas", "\*.txt") if you are working in IntyBASIC and have text files for notes. Leave the comma after "\*.bas", even if you don't have any .txt files.
  - IV. Fourth is the subfolder within the archivepath. If the game FUBAR is backed up to "storage/external-1/Archive/intyBASIC/FUBAR", you would enter "FUBAR/" here.
  - V. The fifth element is an indicator whether your project uses JLP. If so, enter "--jlp"; and if not, enter "". FUBAR does use JLP.
  - VI. The sixth element is the main source file to compile. In all my cases, I use "main.bas" which has Include files for everything else in my projects.

VII. Finally, we have the base filename. For our example, let's say we entered "fubar" here. It is used in several places:

- a. The temporary assembler file (creates "fubar.asm")
- b. The debugger files (creates "fubar.smap", "fubar.sym", and "fubar.lst")
- c. The ROM image files (creates "fubar.bin" and "fubar.cfg")
- d. The box, overlay, instructions, and screenshot files ("fubar.png", "fubar.jpg", "fubar.pdf", and "fubar.gif", in that order, if you have them)

D. When you are done, save and close.

One more thing. I created a .zip file with all the screenshots for the commercial games if you care to download it. It's at the same page as before. Unzip it directly into the folder you specified in the screenshotpath.

So now it's time to go over the bundled files in detail.

I included a keyboard hackfile which was downloaded from <http://www.intellivision.us/intvgames/interface/interface.php> and which I modified. Because Android keyboards don't always have Function keys, which jzIntv uses, I bound some of the jzIntv functions to keys not available on an ECS keyboard. In particular are the three keys to the right of 'P': **J**, **I**, and **N**. Using "PQRS" as a mnemonic, I bound 'P' and the other three keys to the functions: PAUSE, QUIT, RESET, and (screen)SHOT. The **backquote** key (usually to the left of '1') is what I use to toggle between standard controllers and the ECS keyboard, since the ECS keyboard doesn't have that key either. To BREAK to the debugger, I use the **0** key, which is obviously not available when the ECS keyboard controls are in use.

Why '0'? Because I bound the two controller keypads as follows:

123	789
QWE	UIO
ASD	JKL
ZXC	M, .

Which leaves the 0 key free. In addition, the **F** and **H** keys are used to adjust jzIntv's volume.

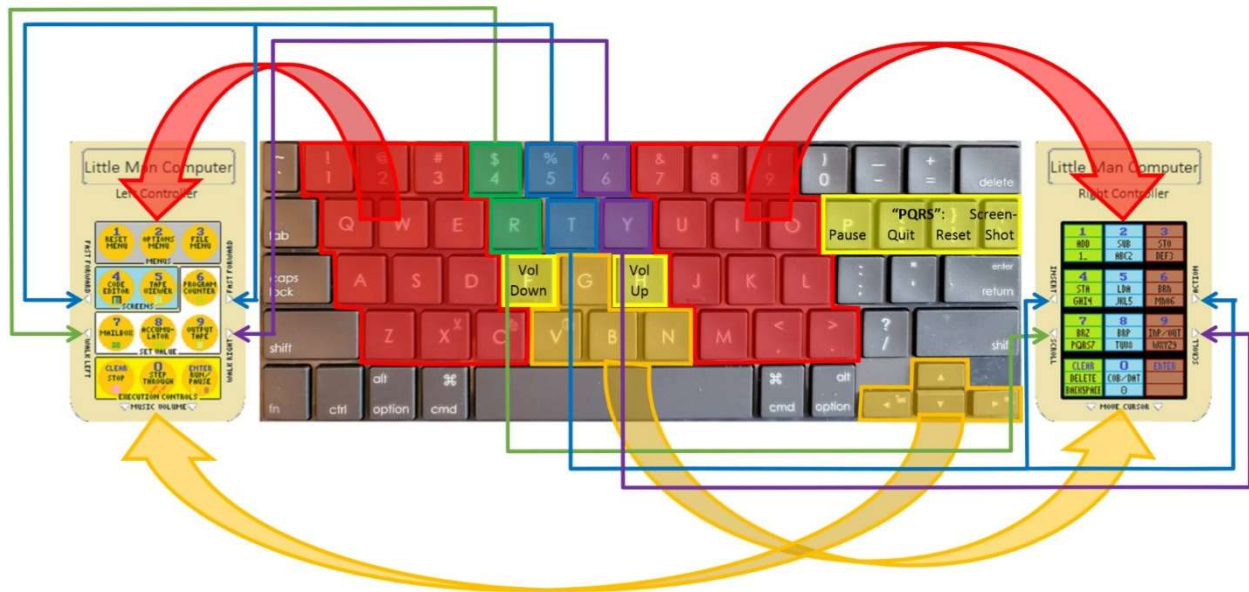
The **arrow keys** are for the left controller directions, and I bound **G**, **V**, **B**, and **N** for the right controller directions. **4**, **5**, and **6** are for the left controller's action keys (bottom left, top, bottom right), and **R**, **T**, and **Y** are for the right controller's action keys. I avoided the Shift, Control, and Alt keys, because most Android keyboards only have one unique key for each of these.

If you don't like it this way, you're free to tweak it to your liking. One thing I should mention is that you can always put the focus on the terminal window and press Ctrl+C to exit, just in case there's trouble with the hackfile.

Because Vectron is unplayable without the "half-wind" directions at the bottom side of the directional disc, I made a separate hackfile for that. Use the **ZXC**, **V**, **B**, and **N** keys for shooting, the **left/right arrows** for moving the Energy Block, and the **up arrow** to toggle Freestyle. All the keypad keys are bound to **123456789-0=** where **=** is Clear and **=** is Enter. That way, you can try to activate the Easter Egg if you know about it.

There are two more files you should know about. One is `intycolors.cfg`, which I used to make “brown” actually look like brown and not olive green, and “light blue” look like light blue and not lavender (or even neon pink). You can tweak that file too if you like. The other file is `dbscript.txt`, which tells `jzIntv` how many characters wide the terminal window is (`jzIntv` is not built to autodetect the terminal width, and the debugger prefers 160 characters or more, which is why I had you set the `geometry` at Step 7E to 160 columns) and starts the game automatically with the debugger enabled.

All the remaining files you don’t need to worry about. The following illustration is part of a document I created to showcase Little Man Computer. I thought it would be helpful to include it here, to help you see how all the keys map to all the Intellivision controls for both controllers. I disabled use of the backquote and 0 keys for the presentation, so they are not illustrated here.



This and the following screenshots are from a previous version of this Portable Intellivision Development Environment, pre-Desktop, to help you appreciate how far it’s come along since then.



## Appendix A: Future Startup and Shutdown Steps

There are exact steps I follow when I want to stop everything for the day.

### Startup

- A. Open Termux.
- B. Type “pkgup” to see if any packages need to be updated.
- C. Type “x”. Disregard any warnings that may start to appear.
- D. Open VNC Viewer.
- E. Select Termux.
- F. Wait until the red, white, and blue panes at the bottom and top of the screen disappear.

### Shutdown

- A. Click on “Applications” and select “Log Out”.
- B. I suggest you uncheck the box to save session for future logins. Click the “Log Out” button.
- C. You will get a message that the server closed unexpectedly. That’s because we wrote the X script to automatically shut down the server as soon as the desktop session is finished. Click/Tap “OK”.
- D. Return to Termux.
- E. Type “exit” as many times as there are active Termux sessions (Termux will close when the last session is exited).

### Occasional

- A. F-Droid and Termux will have new versions available occasionally.
- B. Click on the F-Droid Update page. By default, it will report that all the apps are up to date.
- C. To make sure there truly are no updates, click/tap the Settings page, click on Repositories, uncheck “F-Droid”, and check “F-Droid” again. This will force a repository update.
- D. Now click/tap the Updates page and click/tap “SHOW APPS” if you must.
- E. Make sure you have closed VNC and Termux per the above Shutdown steps before updating Termux.
- F. While updating F-Droid, it will close, naturally. Reopen it when the update is finished.
- G. Re-open Termux and follow the above Startup steps. All your files should be intact.



## Glossary

**API:** Abbreviation of “Application Programming Interface”, a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service. *“Download and install Termux:API in the same manner.”* (Step 1H)

**BASH:** An abbreviation of Bourne-Again SHell, which is the command language interpreter commonly used with Linux. The name is a parody of Bourne Shell, the original Unix shell named after its creator, Stephen Bourne. *“With a bit of tinkering, I put together a text-based environment with a text editor, and some BASH scripts to compile and assemble IntyBASIC code.”* (History, Fall 2018)

**Chmod:** An abbreviation of “CHange MODe”, a command used to change access permissions to files and folders. *“We are going to use the ‘chmod’ command to make the x.sh file executable.”* (Step 3G)

**Diacritic(s):** a sign, such as an accent, which when written indicates a difference in pronunciation. *“The Intellivision does not have diacritics available natively, so I can’t use those for my final ROM titles.”* (Step 10b-C-1a)

**F-Droid:** an app repository that offers FOSS (Free and Open-Source Software) for Android devices. Applications available here are free to download and modify. Because of its nature, F-Droid is not available in Google Play. *“Go to <https://www.f-droid.org> and click/tap on the ‘Download F-Droid’ button.”* (Step 1A)

**Geometry:** the size and position of a window. Most but not all graphical applications allow their dimensions to be defined as a parameter when opening. It can be expressed as pixels or as characters. *“Look for ‘Default geometry’ and change the columns from 80 to 160.”* (Step 7E)

**Half-Wind:** on a 16-point compass rose, the 8 directions between the cardinal and diagonal directions. *“Because Vectron is unplayable without the ‘half-wind’ directions at the bottom side of the directional disc, I made a separate hackfile for that.”* (Step 10b)

**Linux:** an open-source operating system based on Unix. *“You may not know anything about Linux and can’t be bothered to ‘root’ your device.”* (Introduction)

**Man Page:** an abbreviation of “manual page”, part of electronic documentation of a computer software package. *“You might be advised at some point to ‘read the man page for more information’ or something like that.”* (Step 8, “Trying ‘man’”)

**Mnemonic:** (pronounced “nemonic”) a device such as a pattern of letters, ideas, or associations that assists in remembering something. *“Using ‘PQRS’ as a mnemonic, I bound ‘P’ and the other three keys to the functions: PAUSE, QUIT, RESET, and (screen)SHOT.”* (Step 10b)

**Repository/ies:** a place where things are stored, in this case apps or software packages. *“You should see a message about ‘updating repositories’.”* (Step 1E)

**Root** (verb): to gain access to the root account of a device. It is so-called because the root account has full access and can perform functions not authorized by the manufacturer or service provider. There is no need for us to do that here. *“You may not know anything about Linux and can’t be bothered to ‘root’ your device.”* (Introduction)



**Script:** a collection of Linux commands to be executed in sequence. *“With a bit of tinkering, I put together a text-based environment with a text editor, and some BASH scripts to compile and assemble IntyBASIC code.”* (History, Fall 2018)

**SDL:** an acronym for Simple DirectMedia Layer, a cross-platform API that allows software developers to write high-performance computer games and other multimedia applications that can run on multiple operating systems. SDL2 is the newest version. *“In July, jzIntv had a major update to support SDL2 instead of SDL1.”* (History, 2020)

**Symbolic Link:** a type of file that contains a pointer, or reference, to another file or directory. It is sometimes abbreviated “symlink”. *“You will see there are several symbolic links, which Termux colors cyan.”* (Step 21)

**Unicode:** an international encoding standard for use with different languages and scripts, by which each letter, digit, or symbol is assigned a unique numeric value that applies across different platforms and programs. *“You’ll notice in the illustration above that there are Unicode characters to display the titles ‘Kærilon Mini’ and ‘Tron Disc Wars Mêlée’.”* (Step 10b-C-1a)

**Unix:** a widely used multiuser operating system. Its name is a parody of an earlier system called Multics. *“The name [Bourne-Again Shell] is a parody of Bourne Shell, the original Unix shell named after its creator, Stephen Bourne.”* (Glossary, BASH)

**VNC:** an acronym for Virtual Network Computing, a graphical desktop-sharing system to remotely control one computer from another. It sends keyboard and mouse events in one direction, and relays graphical screen updates in the other direction, over a network. *“... Google Play reports that the app VNC Viewer, which we will be using, no longer supports the [Samsung] Tab A.”* (Step 0)



Yet another random picture from my original Environment, pre-Desktop.

## Acknowledgments

These are in no particular order.

- To Óscar Toledo G., for developing IntyBASIC and revolutionizing Intellivision game development, for giving me assistance in getting IntyBASIC to compile in Termux, and for allowing me to contribute “Appendix E” to his latest book, “Advanced Game Programming For Intellivision”.
- To Joe Zbiciak, for developing jzIntv, creating JLP, keeping the old Intellivision Library hosted on his web server, and for all the past and present help with everything from my first attempt at installing Linux in 1999 to getting jzIntv compiled and running in Termux, and finding the Vectron Easter Egg and having enough confidence in me to verify it.
- To “Rev”, who published FUBAR; and to Luc Boudreau, who is now selling FUBAR digitally.
- To Mike Thomasson, for publishing Blix & Chocolate Mine and X-Ray & DILLIGAS.
- To William Moeller, who gave me enough “inside info” in 1998 to make my old website The Intellivision Library relevant, and who kept a fire lit under my feet until I finished FUBAR in 2019.
- To Christopher Neiman, who arranged to have my first two projects put onto cartridge in 2004-2005 (SameGame & Robots), and gave me a Music Synthesizer so I could play Melody Blaster.
- To the Termux community for the obvious reason that my “Intellivision Laboratory” is now mobile, and specifically to those individuals who responded to my various trouble tickets to keep everything working.
- To “JohnPCAE” on AtariAge for developing the Advanced Console Component and sending me a free unit, so I could become one of the first people to develop games for it.
- To “carlsson” on AtariAge for his help in getting syntax highlighting to work for IntyBASIC files in emacs; and to “jenergy” on AtariAge for developing jzIntvImGui, and for helping to port PIDEJL to Windows and taking the remaining screenshots.
- To John W. Shipman at the New Mexico Tech Computer Center for the Tkinter reference manual, which is far better than the documentation on the official Python website.
- Finally, to you, for making it worth my while to have written this document.

The End of this document. The Beginning of your future productivity.

