PRACTICAL PORTS

Let's chat, GPT

BY BENEDICT REUSCHLING

You've probably heard about it by now: there is a new, controversial star in the field of artificial intelligence (AI) called ChatGPT. A lot of things have been written about it, both how it will improve people's lives by doing certain tasks that are considered chores or how it will doom us all by letting the robots win. The spectrum is wide, and the jury is still out on whether we should regulate AI or not. Since this is practical ports, I'm going to take a practical look at what the chatlike tool can do.

I remember my own university classes about artificial intelligence and natural language processing. The former was an unexciting lecture with little value, since, at the time, there were no big breakthroughs happening in the AI field and so the lecture mostly revolved around past efforts in the field—it was a history lesson.

The natural language course in my masters was more engaging. One of the tasks in the class was to program a simple version of Eliza. Back in the day, when the computerized therapist first came out, it was lauded as if it would make psychoanalysts file for unemployment the very next day. But looking at it now, that certainly has not happened. Since we students had to implement this ourselves, we got a better understanding of what the program is actually doing. It was exciting to see what it could do and with a bit of added randomness, it did not get too predictable or boring too soon. The illusion of the almighty, all-knowing program was certainly shattered, having had a look at the inside of Frankenstein's software monster.

> The illusion of the almighty, all-knowing program was certainly shattered.

ChatGPT is obviously more complex and had a lot more knowledge fed into it than Eliza did. The software is impressive but should certainly be taken with a grain of salt as the following experiment demonstrates.

At work, I install cluster machines the old-fashioned way: Boot the FreeBSD ISO image, drop to the shell and set up basic networking (`ifconfig ixl0 up && dhclient ixl0`). Then, I use netcat to transfer a shell script to the host and run that. While there are certainly better and automatic ways (e.g., PXE boot, pre-built images, etc.), this one has served me for a while now. The script is not too complex but has gone through a number of evolutions because of slight variations in the underlying hardware: some nodes have only HDDs, newer ones have NVMe drives, and some have a mixture of HDDs and SSDs. While the script runs fine, I thought I'd let the AI have a look to see if it found any obvious errors or had suggestions for improvements.

The disk setup for the boot, swap, and freebsd partitions for the four disks in the node in combination with ZFS were always a bit tricky. My goal there is to create a RAID-10 zpool (2 mirrors striped together) with the disks having the disk serial numbers as identifiers like this:

```
pool: mypool
 state: ONLINE
config:

NAME                    STATE     READ WRITE CKSUM
mypool                 ONLINE       0     0     0
 mirror-0              ONLINE       0     0     0
   gpt/ABC543210       ONLINE       0     0     0
   gpt/DEF123450       ONLINE       0     0     0
 mirror-1              ONLINE       0     0     0
   gpt/IJK098761       ONLINE       0     0     0
   gpt/LMN678901       ONLINE       0     0     0
```

(See, the gpt is everywhere, but this gpt refers to the GUID partition table). Since the serial numbers are different on each system, I have a little routine that figures out the serial number of each disk and creates a pool, making sure that the mirror keyword is added to the list of serial numbers, as ZFS requires this to create the RAID10.

When I first heard the term GPT, my initial thought was: "why would someone talk to the GUID Partition Table?" I must have worked too much with partitions and disk setups in my day job, so I thought, let's give it a try. Since the GPT in ChatGPT stands for generative pre-trained transformer, the distinction was clear. After creating an account at https://chat.openai.com/chat, a rather unimpressive chat window was presented to me. So, I typed the following:

BCR: How would you create a function for this piece of code:

```
for S in $SSD; do
        gpart add -t freebsd-swap -l swap${COUNTER} -s ${SWAP} -a 4k "${S}"
        echo "Setting ${SWAP} of swap space on disk ${S}."
        SERIAL=$(camcontrol identify "${S}"|awk '/serial number/ { print $3 }')
            gpart add -t freebsd-zfs -l "{$SERIAL}" -a 4k "${S}"

        if [ $((COUNTER%2)) -eq 0 ]; then
                SERIALS="${SERIALS} mirror"
        fi
        SERIALS="${SERIALS} /dev/gpt/${SERIAL}"
        COUNTER=$((COUNTER+1))
        NUM_SSDS=${COUNTER}
done
```

Now, this is neither beautiful code nor elegant, but it works. This is what ChatGPT replied:

ChatGPT: Based on the code snippet you provided, I believe you are trying to create a function that sets up swap and ZFS partitions on multiple disks. Here's a sample function that could accomplish that:

```
setup_partitions() {
    local SSD=$1
    local SWAP_SIZE=$2
    local COUNTER=1
    local SERIALS=""

    for S in $SSD; do
        # Create swap partition
        gpart add -t freebsd-swap -l swap${COUNTER} -s ${SWAP_SIZE} -a 4k "${S}"
        echo "Setting ${SWAP_SIZE} of swap space on disk ${S}."

        # Create ZFS partition
        SERIAL=$(camcontrol identify "${S}"|awk '/serial number/ { print $3 }')
        gpart add -t freebsd-zfs -l "${SERIAL}" -a 4k "${S}"

        # Add to serials and counter
        if [ $((COUNTER%2)) -eq 0 ]; then
            SERIALS="${SERIALS} mirror"
        fi
        SERIALS="${SERIALS} /dev/gpt/${SERIAL}"
        COUNTER=$((COUNTER+1))
    done

    NUM_SSDS=${COUNTER}
}
```

ChatGPT: You can call this function by passing two arguments: an array of SSDs and the desired swap size. For example, if you have two SSDs (`/dev/sda` and `/dev/sdb`) and you want each of them to have 4GB of swap space, you can call the function like this:

```
SSD=("/dev/sda" "/dev/sdb")
SWAP_SIZE="4g"
setup_partitions "$SSD" "$SWAP_SIZE"
```

OK, that's certainly impressive for a start. It could figure out what I was trying to do and offered a solution, along with a fitting explanation. On closer inspection, it needed a couple of modifications: First, the code was written in bash (as the code snippet window was labeled). Since we're on the installer and FreeBSD does not ship with bash by default, bash-specific constructs would not work. Second, it assumed I was using Linux as the disk device nodes indicate (`/dev/sda`). The rest of the code was fine though, and it did not have to modify most of my own code, simply wrapping it into a function.

Now, my prompt (my input into the chat) was not too specific about the system I was running this on, so the AI had to make a best guess about both the shell I was running and the device names. I should have provided this up-front to avoid the AI going in the wrong direction.

What impressed me was how specific the answer was. It felt like I was chatting with a colleague and discussing this code via IRC or other ways of textual communication. This specificity is not to be underestimated. Back when I was studying (I feel old now), the internet was just becoming mainstream, and we still had to go to the library to access certain information. Had I been tasked with a problem like this, I would have had to find a book about shell scripting or ask a fellow student. But in a book, I would get general concepts about programming, explanations, and example code. I would still need to apply what I'd read to the problematic code I had to come up with a fitting solution.

Nowadays, people would google their problem and find similar questions on sites like stackoverflow or forums dealing with coding. I could even post the code there and ask for help if I couldn't find the solution myself. But that requires waiting time for an answer (if anyone actually bothers to reply) and maybe going through a couple of iterations to discuss various approaches.

With tools like ChatGPT, it's like having a personal coding assistant available to discuss your code without a long wait for an answer. Just like Chief Engineer LaForge had "cozy little chats" with the Enterprise's computer, a second pair of eyes can give new ideas and insights that a single person would not have thought of. I could even re-generate the answer that ChatGPT was giving me or ask for more details. Since the AI seems to understand the problem domain (better than a book might), it can make specific recommendations when it is given enough information. Since my colleagues are not available to me 24/7 to discuss a piece of code, I can see that an AI is useful in situations when I need a good answer quickly.

Encouraged, I continued giving the AI a bigger example:

BCR: How would you unify this piece of code into one generic function for both?

```
for HD in $HDD; do
        gpart add -t freebsd-swap -l swap${COUNTER} -s ${SWAP} -a 4k "${HD}"
        echo "Setting ${SWAP} of swap space on disk ${HD}."
        SERIAL=$(camcontrol inquiry "${HD}" -S|sed -e 's/^[[:space:]]*//')

        if [ $((COUNTER%2)) -eq 0 ]; then
                SERIALS="${SERIALS} mirror"
        fi
        SERIALS="$SERIALS /dev/gpt/${SERIAL}"

        gpart add -t freebsd-zfs -l "${SERIAL}" -a 4k "${HD}"

        COUNTER=$((COUNTER+1))
done

COUNTER=0
# we boot from one of the SSDs
for S in $SSD; do
        gpart add -t freebsd-swap -l swap${COUNTER} -s ${SWAP} -a 4k "${S}"
        echo "Setting ${SWAP} of swap space on disk ${S}."
        SERIAL=$(camcontrol identify "${S}"|awk '/serial number/ { print $3 }')
     gpart add -t freebsd-zfs -l "${SERIAL}" -a 4k "${S}"

        if [ $((COUNTER%2)) -eq 0 ]; then
                SERIALS="${SERIALS} mirror"
        fi
        SERIALS="$SERIALS /dev/gpt/${SERIAL}"
        COUNTER=$((COUNTER+1))
        NUM_SSDS=${COUNTER}
done
```

ChatGPT: To unify the two pieces of code into one generic function, you can create a function that takes three arguments: an array of disks, the desired swap size, and a flag indicating whether the disks are HDDs or SSDs. Here's an example function:

```
setup_partitions() {
    local disks=("$@")
    local swap_size=$2
    local is_hdd=$3
    local counter=1
    local serials=""

    for disk in "${disks[@]}"; do
        # Create swap partition
        gpart add -t freebsd-swap -l swap$counter -s $swap_size -a 4k "$disk"
        echo "Setting $swap_size of swap space on disk ${disk}."

        # Determine disk serial number
        if [[ $is_hdd == "true" ]]; then
            serial=$(camcontrol inquiry "$disk" -S | sed -e 's/^[[:space:]]*//')
        else
            serial=$(camcontrol identify "$disk" | awk '/serial number/ { print $3 }')
        fi

        # Create ZFS partition
        gpart add -t freebsd-zfs -l "$serial" -a 4k "$disk"

        # Add to serials and counter
        if [ $((counter%2)) -eq 0 ]; then
            serials="$serials mirror"
        fi
        serials="$serials /dev/gpt/$serial"
        counter=$((counter+1))
    done

    NUM_DISKS=$counter
}
```

ChatGPT: You can call this function with an array of disks, the desired swap size, and a flag indicating whether the disks are HDDs or SSDs. For example, to set up partitions on two HDDs (`/dev/ada0` and `/dev/ada1`) with 2GB of swap space and two SSDs (/dev/ada2 and /dev/ada3) with 4GB of swap space, you can call the function like this:

```
DD=("/dev/ada0" "/dev/ada1")
SSD=("/dev/ada2" "/dev/ada3")
SWAP_SIZE_HDD="2g"
SWAP_SIZE_SSD="4g"

setup_partitions "${HDD[@]}" "$SWAP_SIZE_HDD" "true"
setup_partitions "${SSD[@]}" "$SWAP_SIZE_SSD" "false"
```

ChatGPT: Note that I added a flag to indicate whether the disks are HDDs or SSDs, since the serial number retrieval method is different for each type of disk.

Another impressive response for sure. Especially the verbosity of the reply and explanations beyond handing me some random code snippets. It did figure out the similarities of both functions and could distinguish that the swap size is different for SSDs and HDDs in my example. It was clever enough to figure out the proper disk device nodes (`/dev/ada0` instead of `/dev/sda`). Also, some of my coding style like having uppercase variable names were transformed to lower case. This may or may not be what you like and requires some hand-editing afterwards. The code was still given in bash with some more language specific parts like arrays that are not available in the Bourne shell.

But I rectified my repeated mistake with the following prompt:

BCR: Can you provide the same code in `/bin/sh` only (no bash)?

ChatGPT: Sure! Here's the same code in `/bin/sh` syntax:

```
#!/bin/sh

setup_partitions() {
    disks="$1"
    swap_size="$2"
    is_hdd="$3"
    counter=1
    serials=""

    for disk in $disks; do
        # Create swap partition
        gpart add -t freebsd-swap -l swap$counter -s $swap_size -a 4k "$disk"
        echo "Setting $swap_size of swap space on disk ${disk}."

        # Determine disk serial number
        if [ "$is_hdd" = "true" ]; then
            serial=$(camcontrol inquiry "$disk" -S | sed -e 's/^[[:space:]]*//')
        else
            serial=$(camcontrol identify "$disk" | awk '/serial number/ { print $3 }')
        fi

        # Create ZFS partition
        gpart add -t freebsd-zfs -l "$serial" -a 4k "$disk"

        # Add to serials and counter
        if [ $((counter%2)) -eq 0 ]; then
            serials="$serials mirror"
        fi
        serials="$serials /dev/gpt/$serial"
        counter=$((counter+1))
    done

    NUM_DISKS=$counter
}

HDD="/dev/ada0 /dev/ada1"
SSD="/dev/ada2 /dev/ada3"
SWAP_SIZE_HDD="2g"
SWAP_SIZE_SSD="4g"

setup_partitions "$HDD" "$SWAP_SIZE_HDD" "true"
setup_partitions "$SSD" "$SWAP_SIZE_SSD" "false"
```

ChatGPT: Note that there are a few syntax differences between **`/bin/sh`** and **`/bin/bash`**, such as the use of **=** instead of **==** for string comparison, and the lack of array variables. To work around these differences, I passed the disk lists as space-separated strings, and used **$1**, **$2**, and **$3** to access the function arguments.

So, it was able to transform the code into a working Bourne shell syntax (I tested the results each time).

Now that I had a look at the unified function, I had another idea (which again, could have come from discussing this with a colleague looking over the code):

BCR: What would the code look like when we use the same 4g as the swap size for both HDD and SSD?

ChatGPT: If we want to use the same swap size of 4g for both HDD and SSD, we can modify the **`setup_partitions`** function to take a single disk argument and remove the **`is_hdd`** argument and the conditional logic for determining the disk serial number. Here's what the updated code would look like in **`/bin/sh`** syntax:

```
#!/bin/sh

setup_partitions() {
    disks="$1"
    swap_size="$2"
    counter=1
    serials=""

    for disk in $disks; do
        # Create swap partition
        gpart add -t freebsd-swap -l swap$counter -s $swap_size -a 4k "$disk"
        echo "Setting $swap_size of swap space on disk ${disk}."

        # Determine disk serial number
        serial=$(camcontrol identify "$disk" | awk '/serial number/ { print $3 }')

        # Create ZFS partition
        gpart add -t freebsd-zfs -l "$serial" -a 4k "$disk"

        # Add to serials and counter
        if [ $((counter%2)) -eq 0 ]; then
            serials="$serials mirror"
        fi
        serials="$serials /dev/gpt/$serial"
        counter=$((counter+1))
    done

    NUM_DISKS=$counter
}

HDD="/dev/ada0 /dev/ada1"
SSD="/dev/ada2 /dev/ada3"
SWAP_SIZE="4g"

setup_partitions "$HDD $SSD" "$SWAP_SIZE"
```

ChatGPT: Here, we pass both the HDD and SSD disk lists as a single space-separated string to the `setup_partitions` function and use the same swap size of 4g for both types of disks.

Alright, this is certainly nice as a refactoring step. This time, it did learn that I wanted `/bin/sh` code from my earlier inputs, so it defaulted to it in this next reply.

Bear in mind that results may vary and may or may not work as well as shown here. The AI is constantly evolving, and I surely trained it with my prompts a tiny bit to get better. Each reply had the option to be rated with the familiar thumb-down-or-up buttons.

Next up, I wanted to see what the AI thought about any potential for optimizations:

BCR: Can you recommend other optimizations to the code?

ChatGPT: Sure, here are a few potential optimizations:

1. Use a more efficient way to increment the counter variable: Instead of using `counter=$((counter+1))` inside the loop, you can use the shorthand `counter=$((counter++))` or even just `((counter++))`. This avoids creating a subshell to perform the arithmetic operation, which can be slower.
2. Use the `-n` option for echo to avoid printing a newline character: When printing the message about setting swap space, you can use the `-n` option for echo to avoid printing a newline character, since you're immediately following it with another message. This will make the output look cleaner.
3. Use parameter expansion to remove whitespace: Instead of using `tr -d '[:space:]'` to remove whitespace from the serial number, you can use parameter expansion to achieve the same result: `${variable##*[[:space:]]}`. This avoids the need to spawn an additional process and may be faster.

Here's the code with these optimizations applied:

```
#!/bin/sh

setup_partitions() {
    disks="$1"
    swap_size="$2"
    counter=1
    serials=""

    for disk in $disks; do
        # Create swap partition
        gpart add -t freebsd-swap -l swap$counter -s $swap_size -a 4k "$disk"
        echo -n "Setting $swap_size of swap space on disk ${disk}. "

        # Determine disk serial number
        serial=$(camcontrol identify "$disk" | grep "serial number" | cut -d':' -f2 |
${SED-sed} 's/^[[:space:]]*//')

        # Create ZFS partition
        gpart add -t freebsd-zfs -l "$serial" -a 4k "$disk"

        # Add to serials and counter
        if [ $((counter%2)) -eq 0 ]; then
            serials="$serials mirror"
        fi
        serials="$serials /dev/gpt/$serial"
        ((counter++))
    done

    NUM_DISKS=$counter
}

HDD="/dev/ada0 /dev/ada1"
SSD="/dev/ada2 /dev/ada3"
SWAP_SIZE="4g"

setup_partitions "$HDD $SSD" "$SWAP_SIZE"
```

ChatGPT: These optimizations should make the code slightly more efficient, but keep in mind that the impact on performance may be negligible depending on the size of the disks and the number of partitions being created.

These are certainly some optimizations I had not considered. When running your code through an AI like this, I suggest you approach it with an open mind and leave your programmer ego at the login prompt. Nobody's perfect and the AI may suggest stuff that is either theoretically incorrect or making you fear losing your job because word of the horrible code you've written thus far may reach your boss's ear. But, since this is just between you and the AI, what is stopping you from implementing some of the suggestions to see if they actually improve things?

This is where I find these tools to be a double-edged, blunt sword: if students use it to generate solutions to tasks given to them, no one is stopping them. It's a tool like any other, be it your shell, your compiler, or a handy keyboard shortcut. But, if they can't explain it or just take the code without reflecting on what it does, it becomes dangerous. Cheats don't prosper, especially when they must one day present a solution without any helpers on a whiteboard in front of an examining committee. Understanding is required as well as critical reflection on the solution that is presented. It may have flaws, be inefficient or outright stolen from someone else who provided this exact solution to a forum or stackoverflow and it got fed to the AI's training data. That is a controversy the experts are still discussing.

> It's a tool like any other, be it your shell, your compiler, or a handy keyboard shortcut

Try seeing it as a helper tool—a second pair of artificial eyes looking over your code providing valuable insights. Also, be aware that examiners know about this and may use these tools themselves—either to create assignments or exams that are uniquely tailored to each student.

My own discussion with the AI continued a bit more about parts of this code. I also had it give me suggestions in a separate chat about standing desks of certain dimensions (my workspace is a bit limited) and what optimizations I could put into hadoop configuration files to speed up writes to HDFS.

My suggestion is for people to try it out, take the output with a grain of salt, and use it when it makes sense. I think these tools can improve certain areas like making code more robust or sparking new ideas which then must be implemented by you. I don't anticipate losing my own job because of the existence of these tools any time soon, as there are certain skills that we humans have that are not yet available in ChatGPT and friends.

BENEDICT REUSCHLING is a documentation committer in the FreeBSD project and member of the documentation engineering team. In the past, he served on the FreeBSD core team for two terms. He administers a big data cluster at the University of Applied Sciences, Darmstadt, Germany. He's also teaching a course "Unix for Developers" for undergraduates. Benedict is one of the hosts of the weekly bsdnow.tv podcast.

FreeBSD®

The FreeBSD Project is looking for

- Programmers
- Testers
- Researchers
- Tech writers
- Anyone who wants to get involved

Find out more by

Checking out our website
freebsd.org/projects/newbies.html

Downloading the Software
freebsd.org/where.html

We're a welcoming community looking for people like you to help continue developing this robust operating system. Join us!

Already involved?

Don't forget to check out the latest grant opportunities at
freebsdfoundation.org

Help Create the Future.
Join the FreeBSD Project!

FreeBSD is internationally recognized as an innovative leader in providing a high-performance, secure, and stable operating system.

Not only is FreeBSD easy to install, but it runs a huge number of applications, offers powerful solutions, and cutting edge features. The best part? It's FREE of charge and comes with full source code.

Did you know that working with a mature, open source project is an excellent way to gain new skills, network with other professionals, and differentiate yourself in a competitive job market? Don't miss this opportunity to work with a diverse and committed community bringing about a better world powered by FreeBSD.

The FreeBSD Community is proudly supported by

FreeBSD
FOUNDATION