We can also dump out this op: the current op is always stored in `PL_op`, and we can dump it with `Perl_op_dump`. This'll give us similar output to B::Debug.

```
{
13  TYPE = add  ===> 14
    TARG = 1
    FLAGS = (SCALAR,KIDS)
    {
        TYPE = null  ===> (12)
          (was rv2sv)
        FLAGS = (SCALAR,KIDS)
        {
11          TYPE = gvsv  ===> 12
            FLAGS = (SCALAR)
            GV = main::b
        }
    }
```

finish this later

76.1.18 Patching

All right, we've now had a look at how to navigate the Perl sources and some things you'll need to know when fiddling with them. Let's now get on and create a simple patch. Here's something Larry suggested: if a `U` is the first active format during a `pack`, (for example, `pack "U3C8", @stuff`) then the resulting string should be treated as UTF-8 encoded.

How do we prepare to fix this up? First we locate the code in question - the `pack` happens at runtime, so it's going to be in one of the *pp* files. Sure enough, `pp_pack` is in *pp.c*. Since we're going to be altering this file, let's copy it to *pp.c~*.

[Well, it was in *pp.c* when this tutorial was written. It has now been split off with `pp_unpack` to its own file, *pp_pack.c*]

Now let's look over `pp_pack`: we take a pattern into `pat`, and then loop over the pattern, taking each format character in turn into `datum_type`. Then for each possible format character, we swallow up the other arguments in the pattern (a field width, an asterisk, and so on) and convert the next chunk input into the specified format, adding it onto the output SV `cat`.

How do we know if the `U` is the first format in the `pat`? Well, if we have a pointer to the start of `pat` then, if we see a `U` we can test whether we're still at the start of the string. So, here's where `pat` is set up:

```
    STRLEN fromlen;
    register char *pat = SvPVx(*++MARK, fromlen);
    register char *patend = pat + fromlen;
    register I32 len;
    I32 datumtype;
    SV *fromstr;
```

We'll have another string pointer in there:

```
    STRLEN fromlen;
    register char *pat = SvPVx(*++MARK, fromlen);
    register char *patend = pat + fromlen;
+   char *patcopy;
    register I32 len;
    I32 datumtype;
    SV *fromstr;
```

And just before we start the loop, we'll set `patcopy` to be the start of `pat`: