```
PADOFFSET       pad_add_name(char *name, HV* typestash, HV* ourstash, bool clone)
```

pad_alloc

Allocate a new my or tmp pad entry. For a my, simply push a null SV onto the end of PL_comppad, but for a tmp, scan the pad from PL_padix upwards for a slot which has no name and and no active value.

```
PADOFFSET       pad_alloc(I32 optype, U32 tmptype)
```

pad_block_start

Update the pad compilation state variables on entry to a new block

```
void    pad_block_start(int full)
```

pad_check_dup

Check for duplicate declarations: report any of: * a my in the current scope with the same name; * an our (anywhere in the pad) with the same name and the same stash as `ourstash` `is_our` indicates that the name to check is an 'our' declaration

```
void    pad_check_dup(char* name, bool is_our, HV* ourstash)
```

pad_findlex

Find a named lexical anywhere in a chain of nested pads. Add fake entries in the inner pads if it's found in an outer one. innercv is the CV *inside* the chain of outer CVs to be searched. If newoff is non-null, this is a run-time cloning: don't add fake entries, just find the lexical and add a ref to it at newoff in the current pad.

```
PADOFFSET       pad_findlex(char* name, PADOFFSET newoff, CV* innercv)
```

pad_findmy

Given a lexical name, try to find its offset, first in the current pad, or failing that, in the pads of any lexically enclosing subs (including the complications introduced by eval). If the name is found in an outer pad, then a fake entry is added to the current pad. Returns the offset in the current pad, or NOT_IN_PAD on failure.

```
PADOFFSET       pad_findmy(char* name)
```

pad_fixup_inner_anons

For any anon CVs in the pad, change CvOUTSIDE of that CV from old_cv to new_cv if necessary. Needed when a newly-compiled CV has to be moved to a pre-existing CV struct.

```
void    pad_fixup_inner_anons(PADLIST *padlist, CV *old_cv, CV *new_cv)
```

pad_free

Free the SV at offet po in the current pad.

```
void    pad_free(PADOFFSET po)
```

pad_leavemy

Cleanup at end of scope during compilation: set the max seq number for lexicals in this scope and warn of any lexicals that never got introduced.

```
void    pad_leavemy()
```