SHELL PROGRAMMING QUICK REFERENCE GUIDE

SPECIAL CHARACTERS

;	command separator
( )	execute commands in subshell
{ }	execute commands in current shell
#	comments
$var	variable
&	execute in the background
`	substitute standard out
‘	quote all characters in a string
“	as ‘ but allow substitution

REGULAR EXPRESSIONS

.	match any single character
$	match preceding regular expression at the end of a line
^	match preceding regular expression at the beginning of a line
*	match zero or more occurrences of preceding expression
[ ]	match any character in the brackets (or range, i.e. 2-8)
[^]	match any character not in brackets (i.e., ^0-9 means non-numeric character)
\\	last regular expression encountered
\(exp\)	remember expression for later reference
\{m,n\}	number of times occurring, with m
\{m\}	indicating minimum and n
\{m,\}	indicating maximum

COMMANDS

exit *code*
Exit shell with *code* return code

break *level*
Escape from *level* of for or while loop

continue *level*
Continue from *level* of for or while loop

read
Read input from a file

test
Evaluate an expression or condition

trap
Used for error handling

LOOPING

FOR

```
for variable in file/list
do
        command
done
```

WHILE/UNTIL

```
while/until test/condition
do
        command
done
```

CASE

```
case option in
        option1) command;;
        option2) command;;
        *) command;;
esac
(* is any non-defined option)
```

IF

```
if test/condition then
        command
elif test/expression then
        command
else
        command
fi
```

REPETITION

```
xargs  -n
(see man page for more options)
```

VARIABLE EXPANSION

${var} simple variable substitution
${var:=value}
assign default value if not defined
${var:+value}
substitute value if var is non-null
${var:-value}
temporarily assign value if non-null
${var:?value}
issue error with value if var not set, otherwise substitute value

Compiled by Michael Oliveri (www.mikeoliveri.com)